(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 109 149 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
20.06.2001 Bulletin 2001/25

(51) Int Cl.[7]: **G10H 7/00**, G10H 1/00

(21) Application number: **01200451.1**

(22) Date of filing: **30.11.1995**

(84) Designated Contracting States:
BE GB IT

(30) Priority: **02.12.1994 JP 30002594**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
95308649.3 / 0 715 296

(71) Applicant: **Sony Computer Entertainment Inc.**
Tokyo 107-0052 (JP)

(72) Inventors:
- **Yamanoue, Kaoru,**
  c/o Intellectual Property Depart.
  Tokyo 141 (JP)
- **Okita, Ayako,**
  c/o Intellectual Property Department
  Tokyo 141 (JP)
- **Hashimoto, Takeshi,**
  c/o Intellectual Property Dpt.
  Tokyo 141 (JP)

(74) Representative: **Turner, James Arthur et al**
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

Remarks:
This application was filed on 08 - 02 - 2001 as a divisional application to the application mentioned under INID code 62.

(54) Sound source controlling device

(57) A sound source controlling device in which the processing load required by interpretation of music data may be varied, depending upon the CPU load. The interval of music data interpretation is changed, without changing the music data itself, and the reproduced music composition is not changed in tempo. A system load judgment unit 152 compares the system load information acquired by a system load information acquisition unit 151, with a threshold value stored in a system load threshold value holding unit 153, and accordingly selects a timer interrupt interval held by a timer interrupt interval holder 131. A time information supervisor 143 supervises the acquisition of music paper data held by a music paper data holder, responsive to the timer interrupt interval held by an internal resolution holder 145. A sound enunciation/sound erasure information controller 144 controls a sound source based upon the acquired music paper data.

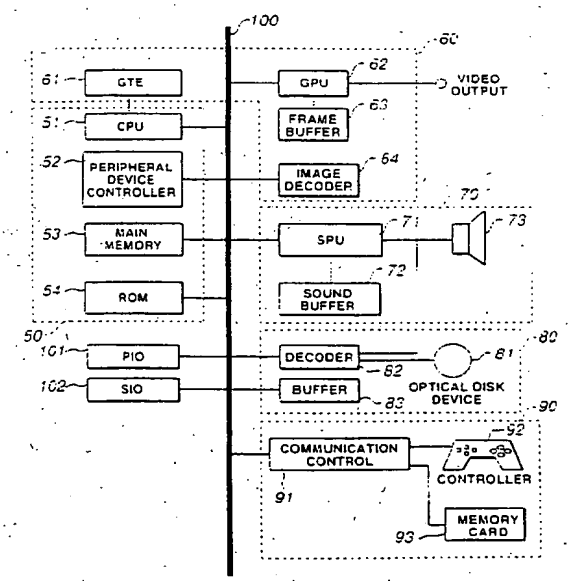


FIG.1

EP 1 109 149 A2

Description

[0001] This invention relates to sound source controlling devices.

[0002] Heretofore, it has been practiced in a video game device or personal computer to generate the music sound or sound effects responsive to the progress of the game or operation by the user.

[0003] In a video game device or a personal computer, a so-called FM music source for changing the frequency of a waveform synthesized from e.g., the fundamental wave and its harmonics for generating a sound with a sound interval, or a PCM music source for holding the waveform of the fundamental wave on memory and changing the read-out period of the fundamental wave responsive to the specified sound interval for generating a sound interval, has been used as a sound source device for generating the sound.

[0004] With such video game device or personal computer, the sound effects generated, the start and stop and the sound volume of the performance of the background music (BGM) can be instantly changed on a real-time basis, responsive to actuation by the user.

[0005] For reproducing the BGM, music paper data, having the interval of the generated sound, sound enunciation and erasure (signifying a rest or pause), and tone color effects arrayed thereon time-sequentially along with the time information, is previously prepared and interpreted on a real time basis for sequentially setting the sound interval, sound enunciation and sound erasure register of the sound source device.

[0006] The method of preparing e.g., BGM data in the form of a music paper data is suited to a multi-media computer game in which the property of promptly responding to the operation by the user plays an important role since the tone color, sound volume or the sound interval can be easily changed during reproduction as compared to a method of sequentially controlling the sound interval, sound enunciation or the sound erasure by program execution.

[0007] For controlling a sound source device based upon the music paper data by a video game device having only a central control unit (CPU) 201 as shown in Fig. 6, the CPU 201 is exploited time-divisionally for reading out music paper data at a pre-set time interval for controlling the sound emission timing, duration of sound emission, interval of the generated sound and the sound volume of a sound source device 202 for generating the BGM.

[0008] With the method of time-divisionally exploiting the CPU 201 for interpreting music paper data, the sound source control device is less costly and the program can be formulated more easily since there is no necessity of providing special peripheral devices provided that the CPU 201 has a sufficiently high processing capability.

[0009] With a game device having a subsidiary CPU 212 for controlling a sound source device 213 in addition to a main CPU 211, as shown in Fig. 7, the subsidiary CPU 212 is used, similarly to the CPU 201, for controlling the sound source device 213 based upon the music paper data.

[0010] With the method of employing the subsidiary CPU 212, the processing for generating the BGM may be carried out completely independently of the operation performed by the main CPU 211 for relieving the load of the CPU 211.

[0011] However, with the method of utilizing the subsidiary CPU 212, the cost of the device is increased since the dedicated subsidiary CPU 212 needs to be annexed for interpreting the music paper data. In addition, since a program for the subsidiary CPU 212 different from that of the main CPU 211 needs to be prepared, the program becomes complex and program formulation becomes difficult.

[0012] With the above-method of time-divisionally utilizing the CPU 201 for interpreting the music paper data, the load of calculation involved in processing other than music paper data interpretation such as picture drawing is increased with the consequence that the processing time of the CPU 201 allocated to the controlling of the sound source device 202 is relatively diminished.

[0013] In this case, if the time interval of executing the sound source control program is changed, the sound generation from the sound source device 202 is delayed, such that the tempo of the generated BGM is changed. For example, if the time interval of execution of the sound source control program is elongated, with the music paper data remaining unchanged, the tempo of the reproduced BGM becomes slower.

[0014] With this in view, a method is usually employed which consists in generating an interrupt at a pre-set time interval of e.g., 1/60 second and control is transferred to the sound source control program by such interrupt. Since this assures that the sound source control program is executed at all times at a pre-set time interval of e.g. 1/60 second, it is possible for the sound control program to effect time control of the music paper data on the basis of such assurance.

[0015] However, with the processing of actual games, it is a frequent occurrence that loads other than the sound source control program such as picture drawing, are increased instantaneously, so that it becomes desirable to prolong the time interval of starting the sound source program for diminishing the processing load of the sound source program.

[0016] In such case, it is necessary to vary the time interval of interrupt generation for starting the sound source control program responsive to the load imposed on the CPU in order to have plural sorts of music paper data for the same BGM in association with the time interval of starting of the sound source control program, thus increasing the volume of music paper data.

[0017] This invention provides audio signal processing apparatus for generating an audio signal by reading out data from a sound source which is controlled by a

central processing unit, comprising:

generator means for generating a plurality of interrupt signals having different interrupt intervals;
detector means for detecting a load condition of said CPU;
selector means for selecting one of said interrupt signals in response to an output of said detector means; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

[0018] This invention also provides a method of generating an audio signal from a sound source which is controlled by a central processing unit, comprising the steps of:

generating a plurality of interrupt signals having different interrupt intervals;
detecting a load condition of said CPU;
selecting one of said interrupt signals in response to the detected load condition of said CPU; and
controlling the read out of data from said sound source in response to said selected interrupt signal.

[0019] Embodiments of the present invention can provide a music sound source device in which the time interval of interpreting music paper data may be changed without changing music paper data and without changing the tempo of the reproduced music composition and in which the processing load of interpretation of the music paper data may be changed responsive to the load imposed on the CPU.

[0020] A sound source control device according to embodiments of the present invention is configured for driving a sound source and for executing information processing other than the sound source control based upon the sound source control information having recorded thereon the control information designed for controlling the sound source along with the time information. The sound source control device includes a sound source control information holder for holding the sound source control information, an interval holder for holding an interval of generating a plurality of timing signal, an interval setter for setting one of the intervals held by the interval holder as an interval of generating a timing signal, a timing signal generator for generating a timing signal at an interval as set by the interval setter, and a sound source controller for reading out the sound source control information corresponding to the interval as set by the interval setter from the sound source control information holder based upon the timing signal from the timing signal generator for controlling the sound source.

[0021] A sound source control device according to embodiments of the present invention includes a load sensor for detecting the information processing load other than the sound source control, and a controller for controlling the interval by the interval setter responsive to a detection output of said load detection unit.

[0022] A sound source control device according to embodiments of the present invention has a picture drawing processing as the information processing other than the sound source control.

[0023] With a sound source control device according to embodiments of the present invention, the time setter sets one of the intervals held by the interval holder as an interval generating a timing signal, and the timing signal generator generates the timing signals at the interval as set by the interval setter.

[0024] The sound source controller reads out the sound source control information conforming to the interval as set by the interval setter from the sound source information holder based upon the timing signals from the timing signal generator and controls the sound source based upon the read-out control information.

[0025] If the interval setter sets one of the intervals of generating the plural timing signals intervals held by the interval holder as a timing signal generating interval, with the interval thus set being other than the currently set interval, the interval of the timing signals generated by the timing signal generator is changed.

[0026] Specifically, the controller controls the setting of the interval setter based upon the load of the information processing other than the sound source control as detected by the load sensor.

[0027] At this time, the load required for controlling the sound source based upon the sound source control information is changed. However, the sound source controller reads out from the sound source control information holder the sound source control information corresponding to a newly set interval of generation of the timing signal and controls the sound source based upon the thus read-out control information.

[0028] Thus, if the interval of generation of timing signals as the reference of the operation of the sound source controller is changed, the sound source control information is read out in accordance with the interval of generation of the changed timing signal, and the sound source is controlled by the thus read-out sound source information.

[0029] The invention will now be described by way of example with reference to the accompanying drawings, throughout which like parts are referred to by like references, and in which:

[0030] Fig. 1 is a block diagram showing the construction of a video game device to which is applied a sound source control device.

[0031] Fig. 2 is a block diagram showing an illustrative construction of a SPU constituting the video game device of Fig. 1.

[0032] Fig. 3 is a block diagram showing the construction of a sound source control device constituting the video game device.

[0033] Fig. 4 illustrates the processing by a sound

controller constituting the sound source device by timer interrupt.

[0034] Fig. 5 illustrates the ratio of the load of the processing operation of the sound controller to that of other processing operations.

[0035] Fig. 6 is a block diagram showing a construction of a conventional sound source control device.

[0036] Fig. 7 is a block diagram showing another construction of a conventional sound source control device.

[0037] An embodiment of the present invention, in which a sound source control device is constituted as a sound source controller foe generating the music sound or the effect sound for e.g. a video game device, is hereinafter explained.

[0038] The video game device, configured for reading out and executing a game program stored in e.g., an auxiliary memory device, such as an optical disc for carrying out the game responsive to instructions from the user, is constructed as shown in Fig. 1.

[0039] That is, the present video game device has a control system 50, composed e.g., of a central processing unit (CPU) and its peripheral devices, a graphic system 60 including a graphic processing unit (GPU) for drawing a picture in a frame buffer, a sound system 70 composed e.g., of a sound processing unit (SPU) for producing e.g., the music sound or the effect sound, an optical disc controller 80 for controlling an optical disc as an auxiliary storage device, a communication controller 90 for controlling a command input from a controller entering a command from the user and input/output to or from an auxiliary storage adapted for storing e. g., game setting, and a bus 100 to which the systems 50 to 90 are connected.

[0040] The control system 50 has a CPU 51, a peripheral device controller 52 for controlling an interrupt or a transfer of a direct memory access (DMA), a main memory 53 composed of a RAM, and a ROM 54 for storage of a program, such as a so-called operating system for supervising the main memory 53, graphic system 60 and the sound system 70.

[0041] The CPU 51 executes the operating system stored in the ROM 54 for controlling the entire device.

[0042] The graphic system 60 has a geometry transfer engine (GTE) 61 for effecting processing such as coordinate transformation, a picture processing device (GPU) 62 for drawing a picture in accordance with picture drawing instructions from the CPU 51, a frame buffer 63 for storing a picture drawn by the GPU 62, and an image decoder 64 for decoding picture data encoded by orthogonal transform such as discrete cosine transform.

[0043] The GPU 62 has a parallel calculation function of executing plural calculations in parallel and is configured for executing coordinate transformation, light source calculations, or matrix or vector calculations at a high speed responsive to demand for calculations from the CPU 51.

[0044] Specifically, for calculations of flat shading of drawing a picture of a triangular polygon to the same color, it is possible for the GTE 61 to effect up to 1,500,000 polygon coordinate calculations a second. Thus it becomes possible with the present video game device to relieve the load imposed on the CPU 51 and to perform high-speed coordinate calculations.

[0045] The GPU 62 carries out drawing of a polygon for the frame memory 62 responsive to a drawing command from the CPU 51. It is possible for the GPU 62 to draw up to a maximum of 360,000 polygons a second.

[0046] This frame buffer 63 is comprised of a so-called dual port RAM and is capable of simultaneously effecting picture drawing from the GPU 62 or transfer from the main memory and readout for display simultaneously.

[0047] This frame buffer 63 has a capacity of 1M bytes and is handled as a matrix of 1024 horizontal pixels by 512 vertical pixels, each pixel being made up of 16 bits.

[0048] An optional area of the frame buffer 63 may be outputted as a video output.

[0049] The frame buffer 63 has, in addition to the display area outputted as a video output, a color look-up table (CLUT) area for storage of the color look-up table to which the GPU 62 refers when drawing e.g., a polygon, and a texture area in which is stored a texture mapped into a polygon drawn by the GPU 62 with coordinate transformation during picture drawing. The CLUT and texture areas are configured for being dynamically changed with changes in the display areas.

[0050] The GPU 62 is able to perform, in addition to the flat shading, the Gouraud shading of deciding the color within the polygon by completing from the color of the apex of the polygon and a texture mapping of affixing the texture stored in the texture area in the polygon.

[0051] When effecting these Gouraud shading or texture mapping, the GTE 61 is able to perform up to 500,000 polygon coordinate calculations a second.

[0052] Under control by the CPU 51, the picture decoder 64 decodes picture data, such as those of a still picture or a moving picture, stored in the main memory 53.

[0053] In addition, the reproduced picture data are stored via the GPU 62 in the frame buffer 63, via the GPU 62, so as to be used as the background for the picture drawn by the GPU 62.

[0054] The sound system 70 has a sound processing unit (SPU) 71 for producing the music sound, effect sound etc. under instructions from the CPU 51, a sound buffer 72 for storing waveform data etc. by the SPU 71 and a speaker 73 for outputting the music sound or the effect sound generated by the SPU 71.

[0055] The SPU 71 has the ADPCM decoding function of reproducing the sound data produced by adaptive differential pulse code modulation (ADPCM) of 16-bit sound data by 4-bit difference signals, a reproducing function of reproducing the waveform data stored in the sound buffer 72 for generating e.g., sound effects, and a modulating function of modulating the waveform data stored in the sound buffer 72 and reproducing the mod-

ulated waveform data.

[0056] By having the above functions, the present sound system 70 can be employed as a so-called PCM sound source for generating the music sound and the sound effects based upon waveform data recorded in the sound buffer 72 under instructions from the CPU 51.

[0057] The optical disc controller 80 has an optical disc device 81 for reproducing the program or data recorded on the optical disc, a decoder 82 for decoding the program or data recorded with e.g., error correction codes, and a buffer 83 for transiently storing playback data from the optical disc device 81 for expediting readout from the optical disc.

[0058] Among the sound data recorded on the optical disc and reproduced by the optical disc device 81, there is the so-called PCM data obtained on analog/digital conversion of sound signals in addition to the above-mentioned ADPCM data.

[0059] The recorded sound data, recorded as ADPCM data by representing the difference of e.g., 16-bit digital data (PCM data) by 4 bits, is decoded by the decoder 82 and subsequently expanded to 16-bit digital data which is then supplied to the above-mentioned SPU 721.

[0060] On the other hand, the sound data, as the PCM data, recorded e.g., as 16-bit digital data, are decoded by the decoder 82 and thence supplied to the SPU 71 or are used for directly driving the speaker 73.

[0061] The controller 90 has a communication control unit 91 for controlling communication with the CPU 51 over bus 100, a controller 92 for entering instructions from the user and a memory card 93 for storing e.g., game setting.

[0062] For inputting the instructions from the user, the controller 92 has e.g., 16 instruction keys and, in accordance with instructions from the communication controller 91, transmits the state of the instruction keys to the communication controller 91 by synchronous communication about sixty times a second. The communication controller 91 transmits the state of the instruction keys of the controller 92 to the CPU 51.

[0063] This enters the instruction from the user to the CPU 51 which then executes processing according to the instructions from the user based upon e.g., the game program which is currently going on.

[0064] If it is necessary to store e.g., the game setting which is currently going on, the CPU 51 transmits data to be stored to the communication controller 91, which then stores data from the CPU 51 in the memory card 93.

[0065] This memory card 93 is connected via communication control unit 91 to the bus 100 and isolated from the bus 100, so that the memory card can be inserted and taken out with the power source turned on. This enables e.g., game setting in plural memory cards 93.

[0066] The present video game device has a parallel input/output (I/O) 101 and a serial input/output (I/O) 102, both of which are connected to the bus 100.

[0067] The video game device may be connected to peripheral equipment via the parallel I/O 101, while it is capable of communicating with other video game devices via serial I/O 102.

[0068] Meanwhile, a large quantity of picture data needs to be transferred among the main memory 53, GPU 62, image decoder 64 and the decoder 82 at the time of reading out the program, displaying or drawing pictures.

[0069] Thus it is possible with the present video game device to effect so-called DMA transfer of directly transferring data among the main memory 53, GPU 62, picture decoder 64 and the decoder 82 under control from the peripheral device controller 52 without interposition of the CPU 51, as explained previously.

[0070] This enables the load on the CPU 51 due to data transfer to be relieved to effect high-speed data transfer.

[0071] With the present video game device, the CPU 51 executes an operating system stored in the ROM 54 when the power is turned on.

[0072] By execution of the operating system, the CPU 51 controls e.g., the graphic system 60 and the sound system 70.

[0073] When the operating system is executed, the CPU 51 initializes the entire device, such as for operation confirmation and then controls the optical disc controller 80 for executing the program of a game etc. recorded on the optical disc.

[0074] By execution of the game program, the CPU 51 controls the graphic system 60 and the sound system 70 responsive to the input from the user for controlling picture display or generation of the effect sound and the music sound.

[0075] Meanwhile, the present video game device has a sound source for generating the sound such as sound effects and a sound source controller for controlling the sound source for generating the music sound or the sound effects with progress of the game, or responsive to the user actuation.

[0076] This sound source is realized by the CPU 51 and the SPU 71, while the sound source controller is realized by the CPU 51.

[0077] Specifically, the SPU 71 has a pitch converter 111 for reading out waveform data recorded in the sound buffer 72 responsive to instructions from the CPU 51 and for converting the pitch of the read-out waveform data, a clock generator 112 for generating clock pulse, a noise generator 113 for generating noise based upon an output of the clock generator 112, a switch 114 for switching between outputs of the pitch converter 111 and the noise generator 113, an envelope generator 115 for adjusting an output of the switch 114 for varying the amplitude of the output waveform for converting the envelope of the produced sound, a muting processor 116 for switching between sound emission or non-emission, and left and right volume control units 117L, 117R for adjusting the sound volume and left and right channel

balance, as shown in Fig. 2.

[0078] The sound buffer 72 has pre-stored therein a plurality of one-period waveform data constituting the sound to be enunciated. These waveform data are stored as the above-mentioned 4-bit ADPCM data and are converted during readout into 16-bit PCM data during readout by the SPU 71 so as to be then supplied to the pitch converter 111.

[0079] Consequently, as compared to the case of directly storing the PCM data, the area within the sound buffer 72 required for storing the waveform data may be diminished for enabling storage of a larger quantity of the waveform data.

[0080] The main memory 53 also has stored therein the envelope of the sound for the one-period waveform data pre-stored in the sound buffer 72, that is the information conceming the sound rise and decay.

[0081] Although a circuit construction for one sound (voice) is shown in Fig. 2, the sound source includes duplicated components from the pitch converters 111 to the volume control units 117L, 117R for a total of 24 sounds (voices). Outputs of the volumes 17L, 17R for the respective vices are synthesized and outputted as the sound output for the left and right channels.

[0082] That is, the sound source is capable of simultaneously enunciating 24 voices.

[0083] The waveform data stored in the sound buffer 72, envelope, sound volume or the balance of the left and right channels may be independently set for the respective voices.

[0084] Thus the sound source is capable of generating chords or performance by plural musical instruments with the use of these voices.

[0085] The sound source is also capable of synthesizing sound outputs with temporal offset by way of effecting a so-called reverberation processing.

[0086] That is, the SPU 71 has switches 118L, 118R for selecting whether or not the sound output synthesized from 24 voices should be reverberated (reverberation processed), a reverberating (reverberation processing) unit 119 of temporally offsetting the sound output supplied from the switch 118L, a volume control unit 120 for adjusting the temporally offset sound volume, an adder 121b for synthesizing an output of the volume control unit 120 to a sound output prior to temporal offsetting, and a master volume unit 122 for adjusting the sound volume of the output of the addition unit 121b.

[0087] The sound source is capable of synthesizing the sound signals read out from the optical disc and supplied from the decoder to the above-described generated sound output.

[0088] Specifically, the SPU 71 has a switch 123 for selecting whether or not the sound signal from the optical disc is to be synthesized to the sound output, a mixing volume control unit 124 for adjusting the sound volume of the synthesized sound signal and supplying the resulting signal to an adder 121a and a switch 125 for selecting whether or not the synthesized sound signal is to be reverberated.

[0089] Although the construction of the reverberating unit 119, volume 120 and the mixing volume 124 is shown in Fig. 2 only with reference to the left channel, the same construction is used for the right channel.

[0090] The operation of the sound source is as follows:

[0091] Whenever the necessity arises for sound enunciation, the CPU 51 supplies a selection signal of selecting the waveform data to be enunciated from among plural waveform data stored in the sound buffer 72, and the sound interval of the sound to be enunciated, to the pitch converter 111, while reading out a envelope corresponding to the waveform data to be enunciated from among the envelopes stored in the main memory 53 and furnishing the read-out envelope to the envelope generator 115.

[0092] The pitch converter 111 varies the waveform data read-out step in accordance with the instructed sound interval in order to read out the waveform data. When the readout of the waveform data for one period comes to a close, the pitch converter 111 iteratively reads out the same waveform data from the outset during the time period the instructions for sound enunciation are issued.

[0093] During the time the instructions for sound enunciation are issued, the waveform data associated with the instructed sound interval is reproduced. These waveform data are supplied via the switch 114 to the envelope generator 115.

[0094] The envelope generator 115 converts the amplitude data of the waveform data from the pitch converter 111 based upon the envelope supplied from the CPU 51.

[0095] This enunciates one-voice sound. The remaining 23 voices of sound are similarly generated and adjusted for sound volume and balance between the left and right channels before being reverberation-processed as described above and synthesized.

[0096] This generates the sound as instructed by the CPU 51.

[0097] Controlling the sound source as described above, is realized by the CPU 51 executing the sound control program.

[0098] The present video game device is so constructed that music paper data having arrayed time-sequentially thereon the music information such as the sound effects to be produced, waveform data used for the background music (BGM), the sound interval of the generated sound, sound enunciation, sound erasure or the tone color is pre-stored along with the time information in the main memory 53 and the sound source controller sequentially reads out the music paper data at a pre-set time interval for sequentially setting the sound interval of the sound source, and the sound enunciation and sound erasure registers for reproducing the effect sound, BGM or the like.

[0099] For controlling the sound source based upon these music paper data, the sound source controller is constructed as shown for example in Fig. 3 illustrating, in an equivalent block diagram, the processing performed by the CPU 51 as a result of execution of the operating system, sound source control program or the game program.

[0100] The sound source controller has a timer interrupt controller 130 for controlling the peripheral device controller 52 for generating timer interrupts to the CPU 51 at a pre-set time interval, a sound controller 140 started at a pre-set time interval by the timer interrupts from the peripheral device controller 52 for controlling the sound source based upon the music paper data, a system load information controller 150 for checking the load state of the video game device in its entirety for supplying the result to the timer interrupt controller 130 and an input demand controller 160 for checking the state of the controller 92.

[0101] As the processing operations simultaneously executed by the CPU 51 with the processing by the sound controller 140 by the execution of the operating system and the game program, there are those executed by a drawing controller 170 for controlling the picture drawing by the graphic system 60 and by a main routine section 180 for selecting the effect sound to be produced, selection of the music sound, selection of the displayed picture and controlling the game process.

[0102] The timer interrupt controller 130 has a timer interrupt interval holder 131 for generating timer interrupts, a timer interrupt supervisor 132 and a control switching supervisor 133 for controlling the switching between the sound controller 140 and the main routine section 180.

[0103] The sound controller 140 has a music paper data holder 141, a data acquisition supervisor 142 for supervising the readout of the music paper data, a time information supervisor 143 for controlling the operation of the data acquisition supervisor 142, a sound enunciation/sound erasure information controller 144 for controlling the sound enunciation/sound erasure based upon the read-out music paper data, an internal resolution holder 145 for holding the internal resolution conforming to the timer interrupt interval from the timer interrupt interval holder 131, and the above sound source.

[0104] The sound source has a sound enunciation section 147 made up of the SPU 71 and the sound buffer 72 and adapted for reading out waveform data stored in a waveform data holder 146 composed of the sound buffer 72 under control by the sound enunciation/sound erasure information controller 144 for generating the sound, and an amplifier 148 for amplifying the produced sound for adjusting the sound volume. The sound enunciation section 147 and the amplifier 148 are realized as performing one of the functions of the SPU 171, as described previously.

[0105] The system load information controller 150 has a system load information acquisition unit 151 for acquiring the system load information, a system load judgment section 152 for judging the system load and a system load threshold value holder 153 for holding the system load threshold value.

[0106] The input request controller 160 has an input 161 made up e.g., of the controller 92, and an input request analysis unit 162 for analyzing the input request from the input device 161.

[0107] The picture-drawing controller 170 has a control time picture drawing information holder 171, made up of the CPU 151, GTE 61, GPU 62 and the frame buffer 63 as well as the GTE 61, a picture-drawing information controller 172, made up e.g., of the CPU 51, a picture-drawing device 173, made up of the GPU 62, a picture-drawing information holder 174, made up of the frame buffer 63, and a display 175 for displaying a picture based upon a video output from the picture-drawing device 173.

[0108] The operation of the sound source controller is as follows:

[0109] With the present sound source controller, the system load or the timer interrupt interval conforming to the input demand is previously held in the timer interrupt interval holder 131. Specifically, the timer interrupt interval for a light system load of 1/240 second, and the timer interrupt interval for a heavy system load of 1/60 second, longer than the value for the light system load, are held in the holder.

[0110] On starting the processing, the sound source controller executes, by the main routine section 180 executed by the CPU 51, the control of the picture-drawing unit 170 responsive to the input from the input device 161, the selection of the music sound generated by the sound controller 140 and processing of the system load information controller 150, as a parallel operation.

[0111] The system load information acquisition unit 151 acquires the load information of the CPU 51 to supply the acquired information to the system load judgment unit 152, which then compares the supplied information to the threshold value held by the system load threshold value holder 153 to judge the system load and transmits the result of judgment to the timer interrupt interval holder 131.

[0112] The timer interrupt internal holder 131 selects the timer interrupt interval, based upon system load judgment from the system load judgment unit 152 or an output of the input request analysis unit 162 and routes the selected interrupt interval to the timer interrupt supervisor 132 and to the internal resolution holder 145.

[0113] Specifically, the timer interrupt interval holder 131 sets the interrupt interval to 1/240 second and to 1/60 second, for the light system load and for the heavy system load, respectively, based upon the result of judgment from the system load judgment unit 152.

[0114] The timer interrupt supervisor 132 controls the peripheral device controller 52, based upon the timer interrupt interval supplied from the timer interrupt interval holder 131, for generating timer interrupt at a pre-set

time interval. The control changeover supervisor 133 switches between processing of the main routine section 180 and the processing of the sound controller 140 at a pre-set time interval based upon the timer interrupt for starting the processing of the sound controller 140.

[0115] When the processing is started with switching of the control changeover controller 133, the time information supervisor 143 of the sound controller 140 controls the data acquisition supervisor 142 responsive to the internal resolution, that is the timer interrupt interval, held by the internal resolution holder 145, for instructing readout of data corresponding to the timer interrupt interval from the music paper data held on the music paper data holder 141, and routes the read-out music paper data to the sound enunciation/sound erasure information controller 144.

[0116] The sound enunciation/sound erasure information controller 144 controls the sound enunciation unit 147 based upon the music paper data supplied from the time information supervisor 143. This causes the sound enunciation unit 147 to generate the sound based upon the waveform data held by the waveform data holder 146.

[0117] Specifically, by execution of the sound enunciation/sound erasure controller 144, the CPU 51 controls the pitch converter 111 and the envelope generator 115 etc. in the manner as described above for controlling the sound generation. The sound thus generated is adjusted in level by the amplifier 148 so as to be outputted by the speaker 73.

[0118] This outputs sound data corresponding to the music paper data for the timer interrupt interval supplied from the timer interrupt interval holder 131.

[0119] The sound controller 140 is started at the timer interrupt interval as set by the timer interrupt interval holder 131 as described above for sequentially generating the sound corresponding to the music paper data for the timer interrupt time interval.

[0120] That is, if the timer interrupt interval is 1/240 second, the music paper data is reproduced at an interval of 1/240 second, as shown in Fig. 4a.

[0121] At this time, the actual processing time of the sound processor 140 is shorter than 1/240 second.

[0122] For example, two music notes are reproduced during time intervals of from time t11 till time t12, from time t12 till time t13, from time t12 till time t14 and from time t14 till time t15. That is, two music notes are reproduced during 1/60 second of from time t11 till time t15.

[0123] If the timer interrupt interval is 1/60 second, music paper data is reproduced at an interval of 1/60 second, as shown in Fig. 4b. For example, eight music notes are reproduced during 1/60 second of from time t21 till time t22.

[0124] That is, eight music notes are reproduced during 1/60 second as in the case of setting the timer interrupt interval to 1/240 second.

[0125] Thus, with the present sound source processing device, even if the timer interrupt interval is changed with the use of the same music paper data, the music paper data is reproduced at a pre-set tempo by controlling the readout of the music paper data responsive to the changed timer interrupt interval.

[0126] If processing is executed by starting the sound controller 140 by interrupt as described above, and the timer interrupt interval is 1/240 second, the processing of the sound processor 140 accounts for 25% of the processing capability of the CPU 51, as shown for example in Fig. 5a. If the timer interrupt interval is 1/60 second, the processing of the sound processor 140 accounts for 12.5% of the processing capability of the CPU 51, as shown for example in Fig. 5b.

[0127] That is, while the load on the CPU 51 for actually controlling the sound source is not vitally changed even if the timer interrupt interval becomes shorter, the overhead for timer interrupt processing is increased if the timer interrupt interval becomes shorter and timer interrupt occurs frequently, so that the load on the sound controller 140 is increased.

[0128] With the above-described sound source controller, the timer interrupt interval selected by the timer interrupt holder 131 is set to 1/240 second and to 1/60 second for the light and heavy system loads, respectively. For the timer interrupt intervals of 1/240 second and 1/60 second, the processing load on the sound controller 140 becomes larger and smaller, respectively.

[0129] Thus, with the present sound source controller, the processing load on the sound controller 140 may be changed responsive to the system load without changing music paper data. Thus, with the heavier system load, the processing load on the sound controller 140 becomes smaller, thus allowing smooth processing, such as picture drawing.

[0130] With the above-described embodiment, the system load judgment section 152 compares the system load information supplied from the system load information acquisition unit 151 to a threshold value held by the system load threshold value holder 153 and selects the timer interrupt interval held by the timer interrupt interval holder 131 based upon the results of comparison. Alternatively, the timer interrupt interval may be controlled by program control of the main routine section 180. Still alternatively, the timer interrupt interval may be set by an input request from the input device 161. The effects similar to those of the above embodiment may be obtained, since it is possible to vary the timer interrupt interval.

[0131] Although the present techniques have been explained in the above embodiment as it is applied to a sound source controller for controlling the sound source in a video game device, the sound source controlling device may be applied to e.g., an automatic performance device or a personal computer provided that the sound source controlling device is adapted for executing other processing operations such as the processing of a picture display device in addition to controlling the sound source. Other modifications may be made if within the scope of the present invention.

[0132] With the sound source controlling device, if the interval of generation of the timing signal as the reference of the operations of the sound source controller is changed by the interval setting device, the sound source control information can be read out in accordance with the interval of generation of the modified timing signals so that the sound source can be controlled by the thus read-out source control information.

[0133] Thus, even if the interval of generation of the timing signals is changed for playback using the same sound source control information without changing the music control information, the music composition can be reproduced at a pre-set tempo so that the tempo of the reproduced music composition is not changed. On the other hand, the load required for controlling the sound source may be changed by changing the interval of generation of the timing signals.

[0134] With the sound source control device, the controller controls the setting of the load of the interval setting unit based upon the load of information processing other than the control of the sound source detected by the load detecting unit so that the load required for controlling the sound source can be changed responsive to the load of information processing other than the control of the sound source and hence the load required for controlling the sound source may be diminished at such time when the load for information processing other than control of the sound source is increased.

[0135] The system load information acquisition unit 151 is implemented by apparatus for determining how busy the main CPU is at any given time. In one arrangement, the system load may be determined by how much time is required for the CPU to service the drawing controller 170. In routines in which the use of the drawing controller is intensive, the main CPU may set a flag before initiating operations having to do with the drawing controller. When these operations are complete, the flag is reset, before other operations are performed in the main routine. This flag may then be examined periodically by an interrupt routine. If the flag is found to be set, it is known that the main routine was interrupted during drawing operations, and if this happens frequently, it is known that there is heavy load on the main CPU because of the drawing operations. In this event, the system load judgment unit recognizes that the system load exceeds the threshold value, and accordingly adjusts the timer interrupt interval held in the timer interrupt interval holder 131. On the other hand, if the drawing flag is found frequently to be reset, it is known that the drawing operations do not represent a heavy load on the main CPU, so that the timer interrupt interval can be adjusted accordingly.

[0136] It will be apparent that various other means can be employed for defining the system load, and for judging whether the system load requires a change in the interrupt interval. These and other modifications of the apparatus disclosed herein may be made by those of ordinary skill in the art, without departing from the central features of novelty of the present invention, which are intended to be defined and secured by the appended claims.

[0137] Embodiments of the invention can thus provide a sound source controlling device in which a music score (paper data) having recorded thereon music information such as the music interval, sound enunciation, rests (sound erasure) or tone color effects of the produced sound in time sequence, is captured at a pre-set interval, and a sound source device is controlled based upon the captured music paper data for automatic performance of e.g., a music composition. More particularly, it relates to a sound source controlling device responsive to the results of calculation or the operation by the user in a video game device or an information processing device for generating the sound effects or background music (BGM).

[0138] In summary, therefore, embodiments of the invention provide a sound source controlling device in which the processing load required by interpretation of music data may be varied, depending upon the CPU load. The interval of music data interpretation is changed, without changing the music data itself, and the reproduced music composition is not changed in tempo. A system load judgment unit 152 compares the system load information acquired by a system load information acquisition unit 151, with a threshold value stored in a system load threshold value holding unit 153, and accordingly selects a timer interrupt interval held by a timer interrupt interval holder 131. A time information supervisor 143 supervises the acquisition of music paper data held by a music paper data holder, responsive to the timer interrupt interval held by an internal resolution holder 145. A sound enunciation/sound erasure information controller 144 controls a sound source based upon the acquired music paper data.

[0139] In so far as the embodiments of the invention described above are implemented, at least in part, using software-controlled data processing apparatus, it will be appreciated that a computer program providing such software control and a storage medium by which such a computer program is stored are envisaged as aspects of the present invention.

[0140] Various respective embodiments of the invention are defined in the following numbered paragraphs:

1. Audio signal processing apparatus for generating an audio signal by reading out data from a sound source which is controlled by a central processing unit (CPU), comprising:

   generator means for generating a plurality of interrupt signals having different interrupt intervals;

   detector means for detecting a load condition of said CPU;

   selector means for selecting one of said interrupt signals in response to an output of said de-

tector means; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

2. An audio signal processing apparatus according to paragraph 1, wherein said CPU also controls a graphic processing apparatus.

3. An audio signal processing apparatus according to paragraph 2, wherein said data read-out from said sound source is musical note data.

4. An audio signal processing apparatus according to paragraph 3, wherein said reading out of data is controlled so that a tempo of said musical note data read out from said sound source is constant regardless of said interrupt signal.

5. An audio signal processing apparatus according to paragraph 3, wherein said sound source comprises a PCM signal.

6. An audio signal processing apparatus according to paragraph 5, wherein said sound source has a waveform memory.

7. An audio signal processing apparatus according to paragraph 3, wherein said sound source comprises a FM signal.

8. A method of generating an audio signal from a sound source which is controlled by a central processing unit (CPU), comprising the steps of:

generating a plurality of interrupt signals having different interrupt intervals;
detecting a load condition of said CPU;
selecting one of said interrupt signals in response to the detected load condition of said CPU; and
controlling the read out of data from said sound source in response to said selected interrupt signal.

Claims

1. An audio signal processing apparatus for generating an audio signal by reading out data from a sound source which is controlled by said audio signal processing apparatus, comprising:

generator means for generating a plurality of interrupt signals having different interrupt intervals;
selector means for selecting one of said interrupt signals based on a load condition of said audio signal processing apparatus; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

2. An audio signal processing apparatus for generating an audio signal by reading out data from a sound source which is controlled by said audio signal processing apparatus, comprising:

generator means for generating a plurality of interrupt signals at an interrupt interval based on an input demand;
selector means for selecting one of said interrupt signals based on a load condition of said audio signal processing apparatus; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

3. An audio signal processing apparatus for generating an audio signal by reading out data from a sound source which is controlled by said audio signal processing apparatus, comprising:

generator means for generating aplurality of interrupt signals having different interrupt intervals;
selector means for selecting one of said interrupt signals ; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

4. An audio signal processing apparatus according to Claim 1, further comprising detector means for detecting said load condition of said audio signal processing apparatus to produce an output based on a result of the detection,
wherein said selector means selects one of said interrupt signals in response to said output of said detector means.

5. An audio signal processing apparatus according to Claim 1,2,3 or 4,
wherein said audio signal processing apparatus also controls a graphic processing.

6. An audio signal processing apparatus according to Claim 5, wherein said data read-out from said sound source is musical note data.

7. An audio signal processing apparatus according to Claim 6, wherein said reading out of data is control-

led so that a tempo of said musical note data read out from said sound source is constant regardless of said interrupt signal.

8. An audio signal processing apparatus according to Claim 6, wherein said sound source comprises a PCM signal.

9. An audio signal processing apparatus according to Claim 8, wherein said sound source has a waveform memory.

10. An audio signal processing apparatus according to Claim 8, wherein said sound source comprises an FM signal.

11. A game machine comprising an audio processing device for generating an audio signal by reading out data from a sound source which is controlled by said audio signal processing device, wherein, said audio signal processing device comprising:

generator means for generating a plurality of interrupt signals having different interrupt intervals;
selector means for selecting one of said interrupt signals based on a load condition of said audio signal processing device; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

12. A game machine comprising an audio processing device for generating an audio signal by reading out data from a sound source which is controlled by said audio signal processing device, wherein, said audio signal processing device comprises,

generator means for generating a plurality of interrupt signals at an interrupt interval based on an input demand;
selector means for selecting one of said interrupt signals based on a load condition of said audio signal processing device ; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

13. A game machine comprising an audio processing device for generating an audio signal by reading out data from a sound source which is controlled by said audio signal processing device, wherein, said audio signal processing device comprises:

generator means for generating a plurality of interrupt signals at an interrupt interval based on an input demand;
selector means for selecting one of said interrupt signals; and
control means for controlling said reading out of data from said sound source in response to said interrupt signal selected by said selector means.

14. A method for making an apparatus having a CPU execute a process for controlling audio source,

said process includes,
a first step in which, in accordance with a load condition of said CPU, an interrupt interval for generating an interrupt signal is changed to switch a control by said CPU to a control processing of said sound source, and
a second step in which said interrupt signal for every interrupt interval changed in the first process is generated.

15. A method according to Claim 14, wherein,
said process further includes a load determination process for determining the load condition of said CPU, and in said first step, the apparatus is made to set said interrupt interval based on a result of the load determination process.

16. A method for making an apparatus having a CPU execute a process for controlling an audio source,

said process includes,
a first step in which, in response to an input demand from an input device, an interrupt interval for generating an interrupt signal is set to switch a control by said CPU to a control processing of said sound source, and
a second step in which said interrupt signal for every interrupt interval set in the first process is generated.

17. A method for making an apparatus having a CPU execute a process for controlling audio source,
said process includes steps of:

generating an interrupt signal for switching said control by said CPU to a control processing of said sound source, and
controlling an interrupt interval for generating said interrupt signal.

18. Computer software comprising program code for carrying out a method according to any one of claims 14 to 17.

19. A storage medium by which computer software according to claim 18 is stored.

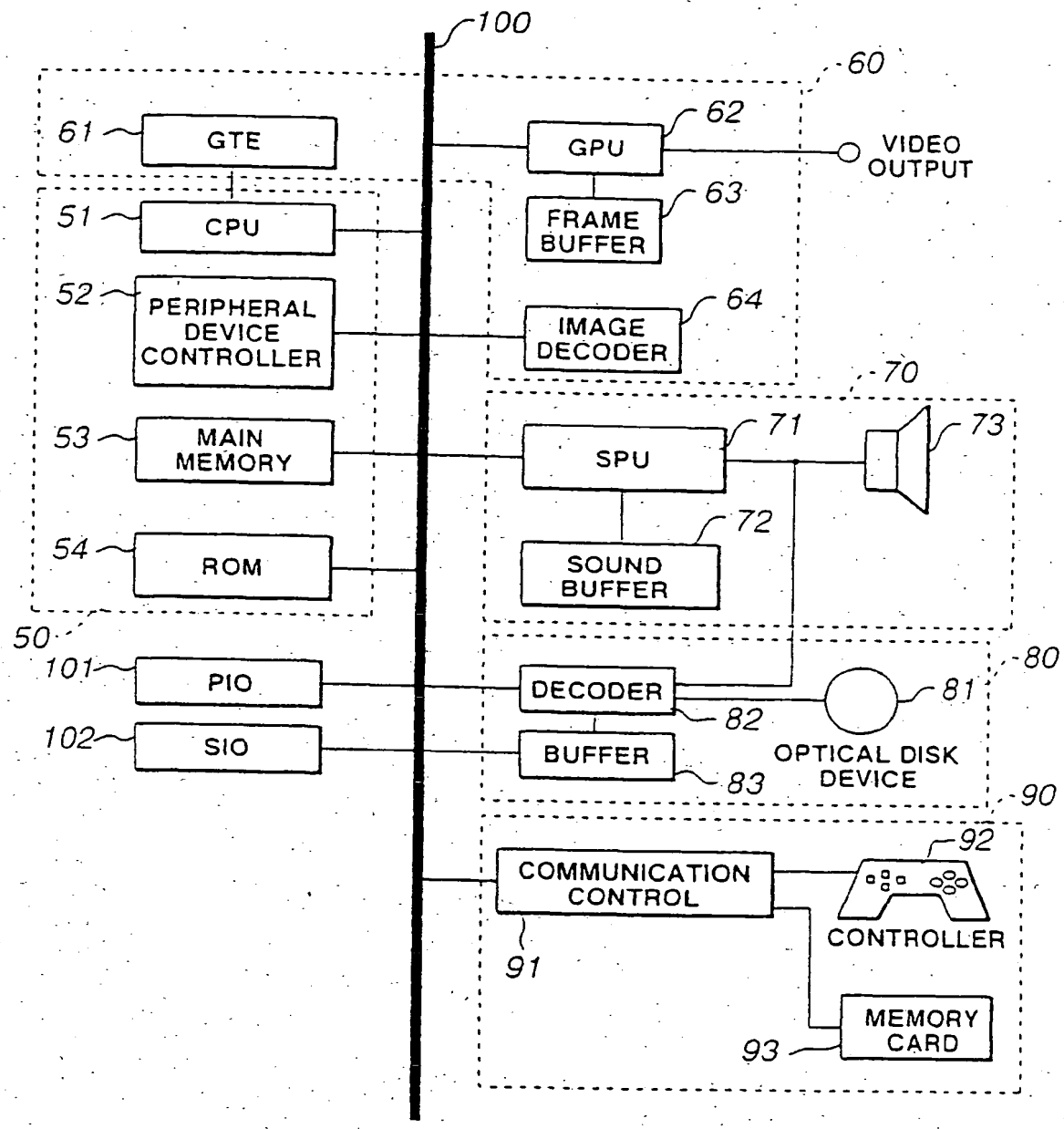


FIG.1

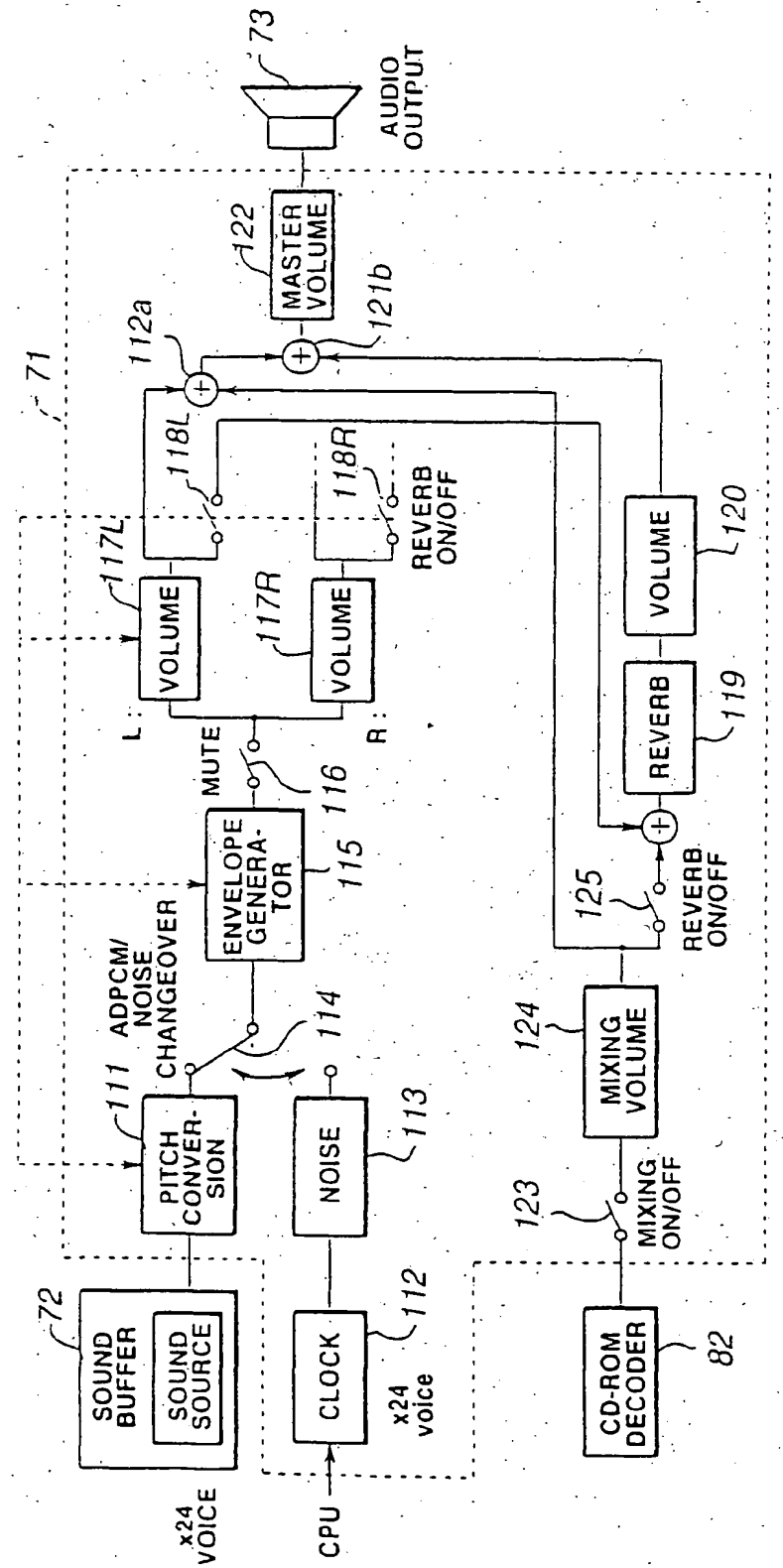


FIG.2

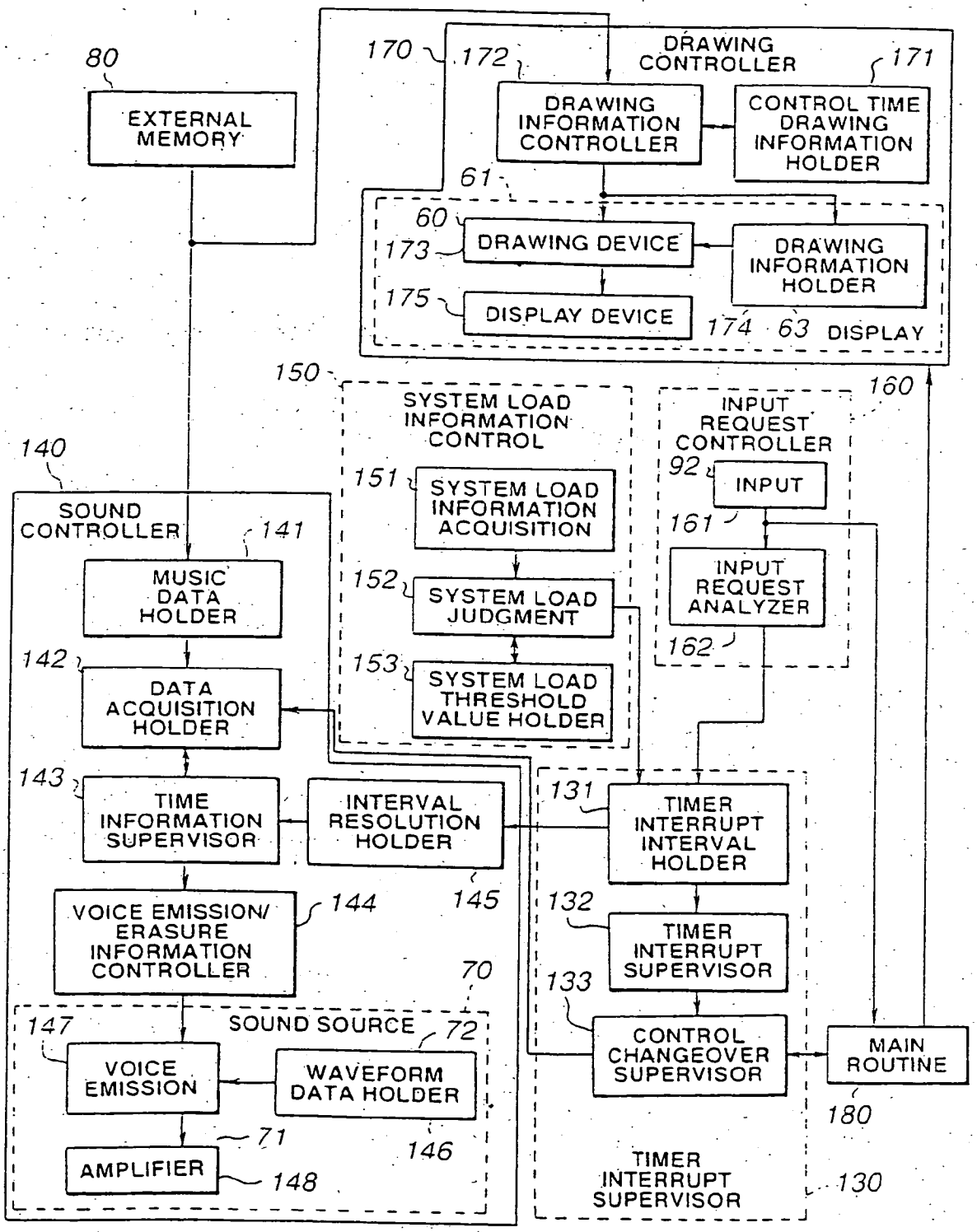


FIG.3

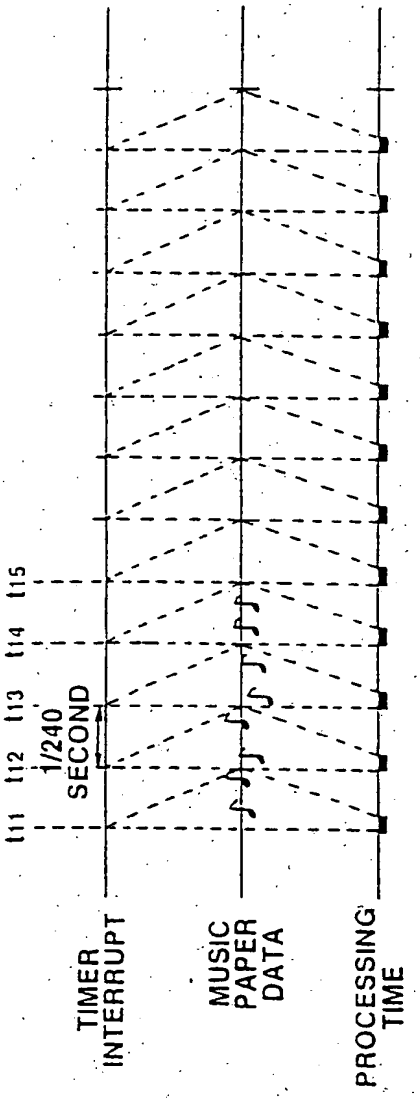


FIG.4A

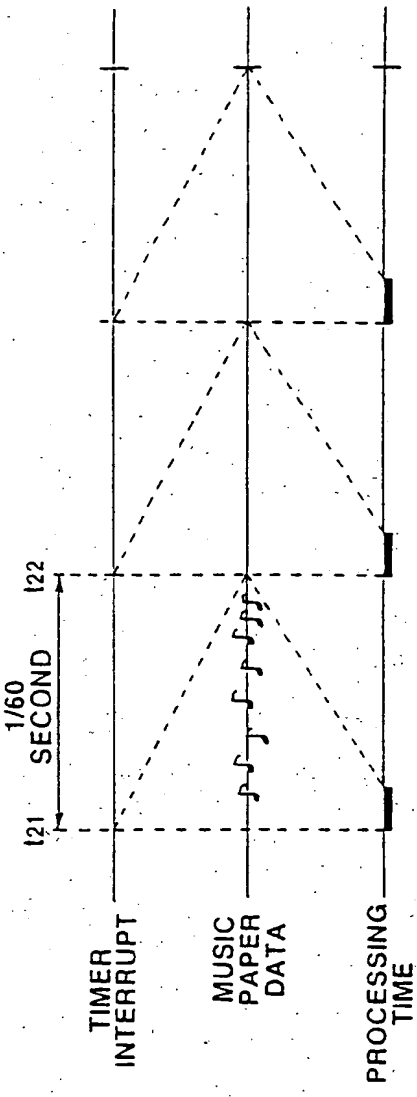


FIG.4B

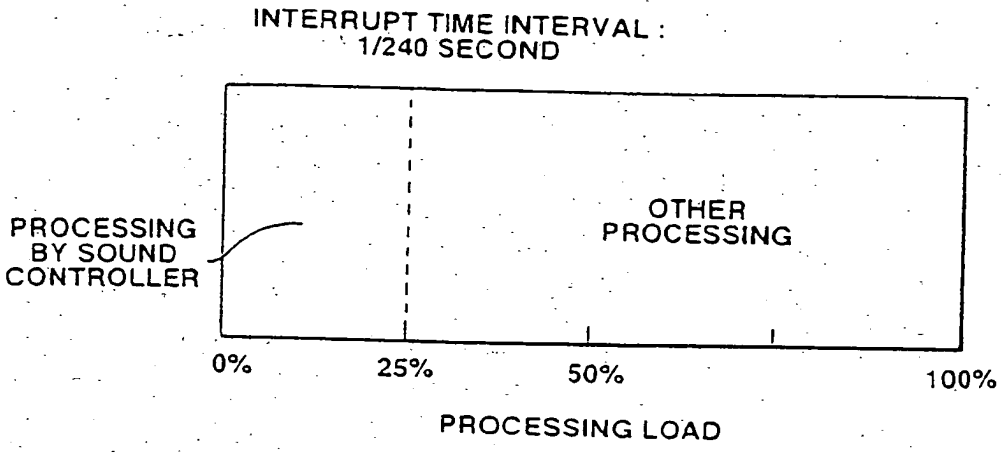


FIG.5A

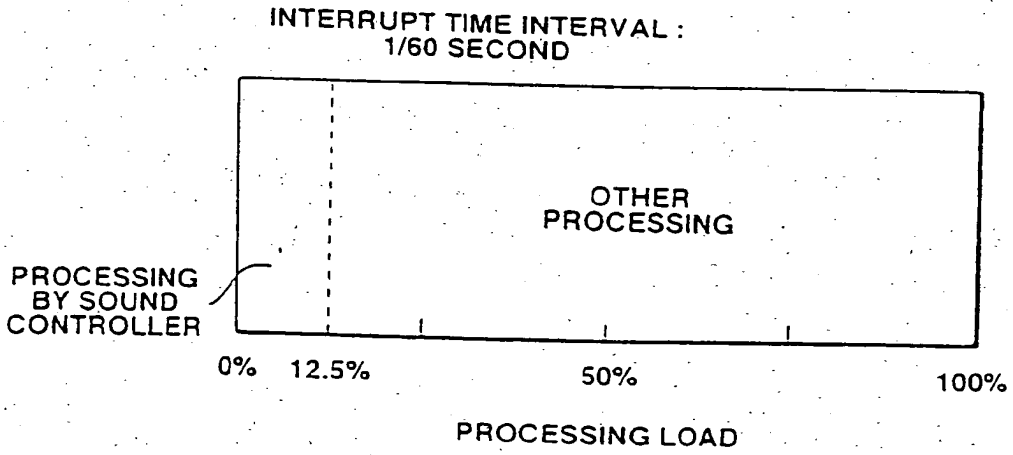


FIG.5B

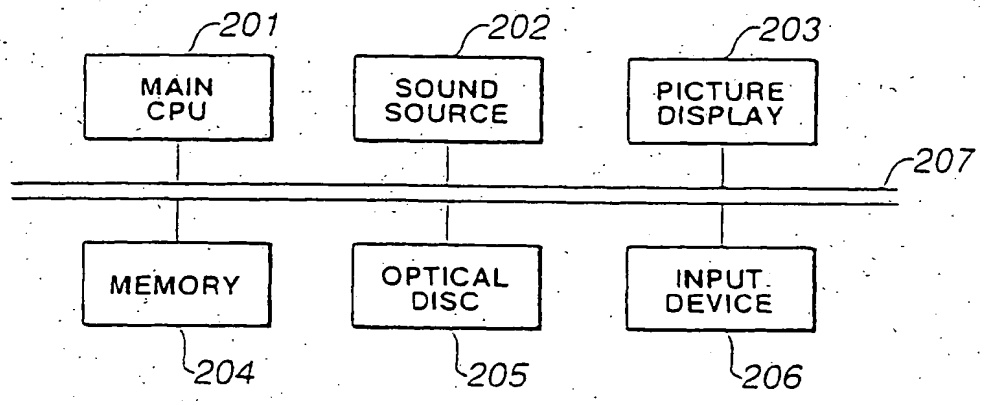


FIG.6

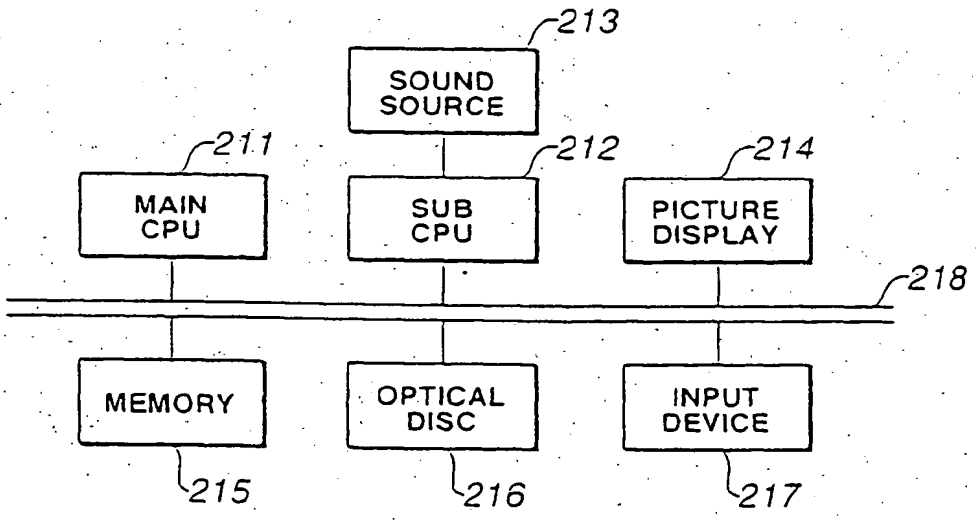


FIG.7

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 109 149 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
18.07.2001 Bulletin 2001/29

(43) Date of publication A2:
20.06.2001 Bulletin 2001/25

(21) Application number: **01200451.1**

(22) Date of filing: **30.11.1995**

(51) Int Cl.7: **G10H 7/00**, G10H 1/00

(84) Designated Contracting States:
BE GB IT

(30) Priority: **02.12.1994 JP 30002594**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
95308649.3 / 0 715 296

(71) Applicant: **Sony Computer Entertainment Inc.**
Tokyo 107-0052 (JP)

(72) Inventors:
- **Yamanoue, Kaoru,**
  c/o Intellectual Property Depart.
  Tokyo 141 (JP)
- **Okita, Ayako,**
  c/o Intellectual Property Department
  Tokyo 141 (JP)
- **Hashimoto, Takeshi,**
  c/o Intellectual Property Dpt.
  Tokyo 141 (JP)

(74) Representative: **Turner, James Arthur et al**
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Sound source controlling device

(57) A sound source controlling device in which the processing load required by interpretation of music data may be varied, depending upon the CPU load. The interval of music data interpretation is changed, without changing the music data itself, and the reproduced music composition is not changed in tempo. A system load judgment unit (152) compares the system load information acquired by a system load information acquisition unit (151), with a threshold value stored in a system load threshold value holding unit (153), and accordingly selects a timer interrupt interval held by a timer interrupt interval holder (131). A time information supervisor (143) supervises the acquisition of music paper data held by a music paper data holder, responsive to the timer interrupt interval held by an internal resolution holder (145). A sound enunciation/sound erasure information controller (144) controls a sound source based upon the acquired music paper data.

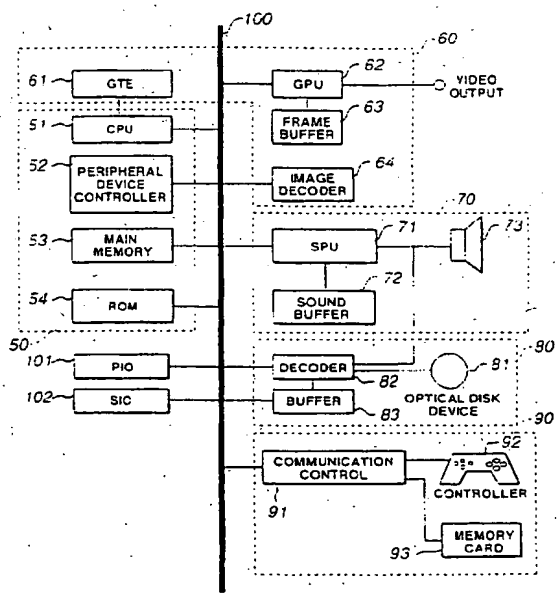


FIG.1

EP 1 109 149 A3

European Patent Office

PARTIAL EUROPEAN SEARCH REPORT

which under Rule 45 of the European Patent Convention shall be considered, for the purposes of subsequent proceedings, as the European search report

Application Number EP 01 20 0451

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	EP 0 597 381 A (IBM) 18 May 1994 (1994-05-18) * page 3, line 21 - page 6, line 17 * * page 9, line 14 - line 30; figures 2,3 *	1-3,5, 16,17	G10H7/00 G10H1/00
A	WO 94 11858 A (MULTIMEDIA SYSTEMS CORP) 26 May 1994 (1994-05-26) * page 7, line 3 - page 8, line 6 * * page 28, line 5 - line 22; figures 2A,2B,9 *	1-3,5,6, 11-14, 16,17	
A	EP 0 463 411 A (CASIO COMPUTER CO LTD) 2 January 1992 (1992-01-02) * column 14, line 1 - column 16, line 30; figures 2,4,5 *	1-3,11, 14	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

G10H

INCOMPLETE SEARCH

The Search Division considers that the present application, or one or more of its claims, does/do not comply with the EPC to such an extent that a meaningful search into the state of the art cannot be carried out, or can only be carried out partially, for these claims.

Claims searched completely :

1-17

Claims searched incompletely :

Claims not searched :

18,19

Reason for the limitation of the search:

Article 52 (2)(c) EPC - Program for computers

Place of search	Date of completion of the search	Examiner
THE HAGUE	27 April 2001	Pulluard, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C07)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 20 0451

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

27-04-2001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0597381 A	18-05-1994	JP 7146679 A	06-06-1995
		US 5515474 A	07-05-1996
WO 9411858 A	26-05-1994	AU 5212098 A	19-03-1998
		AU 682836 B	23-10-1997
		AU 5602294 A	08-06-1994
		BR 9307440 A	01-06-1999
		CA 2148089 A	26-05-1994
		EP 0669037 A	30-08-1995
		JP 8503584 T	16-04-1996
		NZ 258384 A	24-06-1997
EP 0463411 A	02-01-1992	JP 2869573 B	10-03-1999
		JP 4058291 A	25-02-1992
		JP 2869574 B	10-03-1999
		JP 4060698 A	26-02-1992
		DE 69130748 D	25-02-1999
		DE 69130748 T	30-09-1999
		HK 1013349 A	23-06-2000
		KR 9500841 B	02-02-1995
		US 5726371 A	10-03-1998
		US 5319151 A	07-06-1994

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 951 009 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
20.10.1999 Bulletin 1999/42

(51) Int. Cl.[6]: **G10H 7/00**

(21) Application number: **99112006.4**

(22) Date of filing: **03.06.1996**

(84) Designated Contracting States:
DE GB IT

(30) Priority: **06.06.1995 JP 13952695**
29.09.1995 JP 25349395

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
96108875.4 / 0 747 877

(71) Applicant: **YAMAHA CORPORATION**
Hamamatsu-shi, Shizuoka-ken 430 (JP)

(72) Inventors:
- **Wachi, Masatada**
  Hamamatsu-shi, Shizuoka-ken,430 (JP)
- **Yamada, Hideo**
  Hamamatsu-shi, Shizuoka-ken,430 (JP)
- **Hirano, Masashi**
  Hamamatsu-shi, Shizuoka-ken,430 (JP)

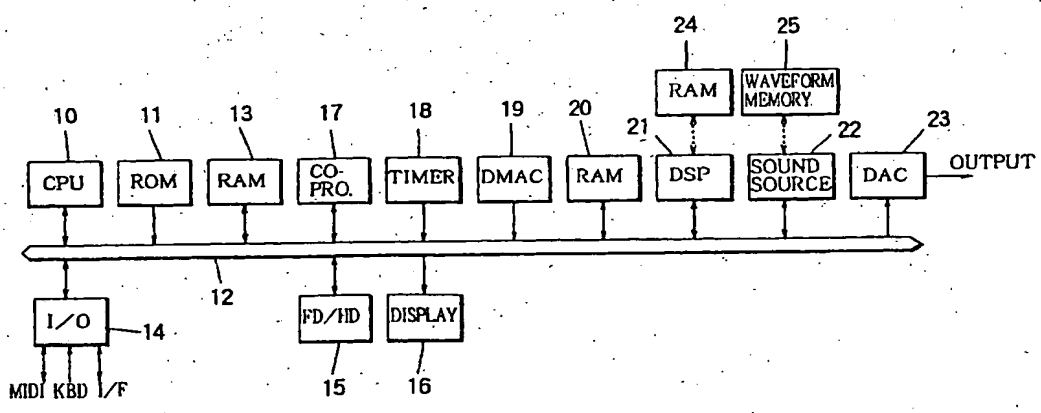
(74) Representative:
Kehl, Günther, Dipl.-Phys.
Patentanwaltskanzlei
Günther Kehl
Friedrich-Herschel-Strasse 9
81679 München (DE)

Remarks:
This application was filed on 21 - 06 - 1999 as a divisional application to the application mentioned under INID code 62.

(54) Computerized music system having software and hardware sound sources

(57) Disclosed is a music apparatus having a central processor, a method and a machine readable medium for use in a music apparatus, all of which are designed for generating a musical sound according to performance information. In the apparatus, a waveform generator creates digital waveforms. It is composed of a software program which can be executed by the central processor in a variable operation mode. This variable operation mode is changed according to the computation capacity available for operation of the waveform generator. The musical sound is then generated based on the created digital waveform.

FIGURE 1



EP 0 951 009 A1

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to a musical sound generator employed in computer application systems using CPU, such as electronic game machines, network karaoke apparatuses, personal computers etc., and more particularly relates to a musical sound generator capable of generating various musical sounds according to performance information.

[0002] Conventionally, in an apparatus such as a personal computer, musical sound can be reproduced by means of a specialized hardware module such as a sound source LSI and a sound source board, and by executing programs to control the installed sound source device. Recently, performance of CPU in the personal computer is remarkably improved so that the musical sound can be generated by the CPU in place of the specialized hardware module. This sort of the musical sound generation is called "software sound source" in contrast to "hardware sound source" which generates musical sound by the specific hardware. The CPU computes waveform data of musical sound according to a specific program. The quality of the sound generated by the software sound source depends upon the performance of the CPU executing the program. If the CPU has high performance, the waveform data can be computed at high speed so that a sampling frequency of the waveform data can be raised high to realize high quality musical sound generation. However, if the CPU performance is poor, it is difficult to compute the data at high speed so that the sampling frequency must be lowered. This inevitably results in poor quality of the reproduced musical sound.

[0003] From system to system, there is a wide variety in the configuration of the application systems such as the personal computer, with respect to installation of optional devices. Generally, the optional device includes a hard disk, a video card etc., in broader meaning. However, in the present invention, the optional device means an externally connectable device involved in the musical sound generation. It should be noted that the processing ability of the CPU may be different in various system configurations. Further, in a situation where an application program to generate sound and another application program to execute other jobs are simultaneously invoked in parallel, a load of the CPU may vary dependently on the running status of the programs and on the status of tasks currently executed in the system. In this fashion, the effective processing power of the CPU may vary in the same system. Thus, a user must rearrange basic setup for the sound generation whenever an environment of the system is changed, and that is very laborious. In such systems, the user cannot recognize whether the sound is not generated correctly in a current configuration setup, until any miss and skip of tone occurs at the actual reproduction of the sound. In other words, it is impossible to evaluate whether the system configuration is reasonable until the sound is actually reproduced. Further, no matter how the CPU performance is high, an external sound generation hardware may be utilized in many cases according to the user's desire in practical use of the hardware resource in such cases, full use of the hardware sound source may cause a problem that it is impossible to generate a sound surpassing the limitation of the facility of the sound generation hardware. A variety of tones cannot be reproduced. A number of timbre kinds cannot be increased even if there is a, sufficient processing power in the CPU.

SUMMARY OF THE INVENTION

[0004] The purpose of the present invention is to solve the problems described above, and is to provide a musical sound generator capable of generating various musical sounds with retaining excellent quality of the sounds.

[0005] According to the invention, a music apparatus having a central processor for generating a musical sound according to performance information comprises means for receiving the performance information; a waveform generator composed of a software program executable by the central processor in a variable operation mode dependently on a computation capacity of the central processor to create a digital waveform; means for changing the variable operation mode of the waveform generator according to the computation capacity available for operation of the waveform generator; the central processor operating the waveform generator in the variable operation mode as changed to create the digital waveform according to the received performance information; and means for generating the musical sound based on the created digital waveform.

[0006] Specifically, the waveform generator is operable in the variable operation mode having a variable operation speed to create a digital waveform by successively computing sample values of the digital waveform, and is provisionally operated to carry out trial creation of a model digital waveform while measuring the computation capacity of the central processor in terms of the operation speed at which the trial creation is carried out, wherein the means for changing comprises means for determining the variable operation mode in terms of a sampling frequency comparable to the measured operation speed, and wherein the central processor actually operates the waveform generator to enable the same to successively compute sample values of the digital waveform at the determined sampling frequency.

[0007] The music apparatus may include means for detecting the computation capacity available for operation of the waveform generator by detecting whether or not an additional processor is available to assist the central processor in computation for executing the software program. The additional processor may also comprise a co-processor of the central processor.

[0008] The music apparatus according to the invention may include means for detecting the computation capacity available for operation of the waveform generator by provisionally measuring the computation capacity of the central processor before the central processor executes the software program to operate the waveform generator.

[0009] The means for changing may comprise means for changing the variable operation mode such that a first algorithm defining a method of creating the digital waveform is changed to a second algorithm simpler than the first algorithm when the computation capacity of the central processor decreases.

[0010] The means for changing may comprise means for changing the variable operation mode according to the computation capacity of the central processor in terms of a variable sampling frequency by which the waveform generator variably creates samples of the digital waveform.

[0011] The means for changing may comprise means for changing the variable operation mode such that a set of computation steps performed by the central processor to create the digital waveform is changed according to the computation capacity of the central processor.

[0012] The invention also relates to a method of generating a musical sound by a central processor according to performance information. According to the invention, the method comprises the steps of: receiving the performance information; preparing a waveform generator composed of a software program executable by the central processor in a variable operation mode dependently on a computation capacity of the central processor to create a digital waveform; changing the variable operation mode of the waveform generator according to the computation capacity available for operation of the waveform generator; operating the waveform generator in the variable operation mode as changed to create the digital waveform according to the received performance information; and generating the musical sound based on the created digital waveform. Furthermore, the invention relates to machine readable medium for use in a music apparatus having a central processing unit for generating a musical sound according to performance information. The medium contains instructions executable by the central processing unit for causing the music apparatus to perform a method comprising the steps of: receiving the performance information; preparing a waveform generator composed of a software program executable by the central processing unit in a variable operation mode dependently on a computation capacity of the central processing unit to create a digital waveform; changing the variable operation mode of the waveform generator according to the computation capacity available for operation of the waveform generator; operating the waveform generator in the variable operation mode as changed to create the digital waveform according to the received performance information; and generating the musical sound based on the created digital waveform.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013]

Figure 1 is a schematic block diagram illustrating a first embodiment of the inventive musical sound generator.

Figure 2 is a schematic block diagram illustrating a variation of the embodiment shown in Figure 1.

Figure 3 is a schematic block diagram illustrating another variation of the embodiment shown in Figure 1.

Figure 4A is a schematic block diagram illustrating an arrangement in which a sound source device and its peripheral devices are integrated with each other.

Figure 4B is a schematic block diagram illustrating an arrangement in which a DSP and its peripheral devices are integrated with each other.

Figure 5 illustrates operating modes in the first embodiment according to the present invention.

Figure 6 is a memory map of a RAM provided in the first embodiment.

Figure 7 is a flowchart illustrating an overall process executed in the first embodiment.

Figure 8 is a flowchart illustrating a waveform synthesizing program executed in the first embodiment.

Figure 9 is a flowchart illustrating the waveform synthesizing program executed in the first embodiment.

Figure 10 is a flowchart illustrating the waveform synthesizing program executed in the first embodiment.

Figure 11 is a flowchart illustrating the waveform synthesizing program executed in the first embodiment.

Figure 12 is a flowchart illustrating the waveform sample value loading process executed in the waveform synthesizing program.

Figure 13 is a flowchart illustrating the waveform sample value computation process executed in the waveform synthesizing program.

Figure 14 is a flowchart illustrating the waveform sample value computation executed by CPU in the waveform synthesizing program.

Figure 15 is a flowchart illustrating the synthesizing process by a selected hardware in the waveform synthesizing program.

Figure 16 is a flowchart illustrating the synthesizing process by the selected hardware in the waveform synthesizing program.

Figure 17 is a flowchart illustrating a timer process in the waveform synthesizing program.

Figure 18 is a flowchart illustrating a synthesizing process by a selected hardware in a second

embodiment.

Figure 19 is a flowchart illustrating a synthesizing process by a selected hardware in a third embodiment.

Figure 20 is a flowchart illustrating a synthesizing process by a selected hardware in a fourth embodiment.

Figure 21 is a schematic block diagram illustrating a variation to which the present invention is applied.

Figure 22 is a schematic block diagram illustrating another variation to which the present invention is applied.

Figure 23 is a schematic block diagram illustrating a further variation to which the present invention is applied.

Figure 24 is a schematic block diagram illustrating a still further variation to which the present invention is applied.

Figure 25 is a schematic block diagram showing an additional embodiment of the inventive musical sound generating apparatus.

DETAILED DESCRIPTION OF THE INVENTION

[0014] Details of embodiments of the present invention will be described hereunder with reference to the drawings. Figure 1 is a schematic block diagram showing a first embodiment of a musical sound generator according to the present invention. In Figure 1, numeral 10 denotes a CPU, which controls various units composing a computer system via a data bus 12; according to a basic program stored in a ROM 11. Numeral 13 denotes a RAM temporarily storing various registers, flags and data. Numeral 14 denotes an I/O port of the multi type which receives and transmits MIDI information, key information provided upon key operation of a keyboard (not shown), and other miscellaneous information via various interfaces I/F (not shown). The multi type I/O port 14 receives performance information in the form of the MIDI information or the key information KBD. In the present embodiment, it is assumed that the performance information may be generated by an automatic performance program. In this case, the automatic performance program means that the performance information is generated in time series by a certain automatic sequence program. Therefore, the arrangement shown in Figure 1 works not only as a musical sound generator but also a sequencer. The type of I/F may be a serial or parallel port, RS-232C, RS-422 and so on. Especially in case of RS-232C, the computer system communicates with a host station through a public telephone network by a modem (not shown). Thus, the input source of the performance information may be the keyboard in case that the keyboard operation information is provided, or an external device connected through the I/F in case that the MIDI information is provided, or a sequence program executed by the CPU in case that the automatic performance information is provided. Numeral 15 denotes a storage unit, which is comprised of FD (Floppy Disk) or HD (Hard Disk). The storage unit 15 further stores application programs and data. Numeral 16 denotes a display which is composed of CRT or LCD (Liquid Crystal Display). The display 16 presents various data under the control of the CPU 10. Numeral 17 denotes an optional co-processor which executes floating point computation instead of the CPU 10. The rest of data processing is carried out by the CPU 10. Numeral 18 denotes a timer which counts time in timer processing described later. Numeral 19 denotes a DMAC (Direct Memory Access Controller) which transfers data to and from RAM 20 directly without the control by CPU. Nowadays, the co-processor 17, timer 18 and DMAC 19 may be accommodated in a single chip together with CPU 10, though these are discrete devices in the present embodiment. Numeral 20 denotes a RAM which has a structure similar to the first-mentioned RAM 13 with respect to its hardware construction. However, the RAM 13 is used as a work area for the program execution by the CPU, while the other RAM 20 is a waveform memory which temporarily stores a waveform represented by wave data. Numeral 21 denotes a DSP (Digital Signal Processor) for use in digital signal processing required for musical sound synthesis. Numeral 22 denotes an optional sound source which constitutes a first waveform generator of one chip LSI for generating a waveform of the musical sound according to performance information. Numeral 23 denotes a D/A converter which is enabled when a flag DACENBL is set to "1". Before the D/A converter 23, an FIFO data buffer (not shown) is normally provided. The wave data stored in the FIFO is read out at a sampling frequency fs. After the D/A converter 23, an LPF (Low Pass Filter) is normally provided (not shown). A cutoff frequency of the LPF is set to about half of the sampling frequency fs. The LPF is an output device of the musical sound generator. The musical sound is reproduced through an amplifier and a speaker. Numeral 24 denotes a RAM which is structured similarly to the RAM 13 or 20 with respect to its hardware construction. The RAM 24 is utilized as a work memory for the arithmetic operation of the DSP 21. Numeral 25 denotes a waveform memory which stores wave data of basic or typical timbres in case that the sound source 22 generates a musical sound according to a waveform memory readout method. The role of the waveform memory 25 and the RAM 20 is slightly different in a manner that the waveform memory 25 is used mainly by the sound source 22 and is provided in the form of ROM or a daughter board, while the RAM 20 is utilized by the CPU 10 as a waveform memory.

[0015] Generally in the arrangement above, some devices such as co-processor 17, DSP 21, sound source 22, RAM 24, and waveform memory 25 are often optionally installed according to user's choice. If the RAM 24 is not installed, a certain area of the RAM 13 is allocated for the DSP 21. The waveform memory 25

may not be installed if the sound source 22 is an FM synthesizing device which generates a musical sound by pure computation of wave data. The CPU 10 recognizes whether these optional devices are installed or not. In the present embodiment, the CPU 10 recognizes the existence of the optional devices according to one of methods listed below:

(1) A port address is reserved for a corresponding device connection. The CPU 10 accesses the port address immediately after power-on or reset of the system. If the CPU 10 detects a predetermined sign from the port address, it recognizes the existence of the corresponding device.

(2) A jumper switch is provided for signifying the device installation. The user turns the switch on at the time of installing the corresponding device.

(3) If the system is implemented as a personal computer, the optional devices are registered in a configuration file in terms of corresponding device drivers or in a batch file. The system software recognizes the devices through these files.

[0016] All these optional devices are fully installed and connected to the data bus 12 in the present embodiment. However, the connection port is not limited to the data bus 12. The optional devices may be connected via serial/parallel interfaces, through which each device carries out data transaction mutually with the CPU 10. In other words, any kind of interface is possibly provided for the optional devices, provided that the device can communicate with the CPU through the interface. For example, the sound source 22 may actually be provided as a sound source board 41 or an extension board as shown in Figure 2. In this case, the board 41 is inserted into a slot on a main board or mother board. The sound source board 41 communicates with the CPU 10 through the bus 12, I/F controller 26, and extension interface 27. The waveform memory 25 may be installed through a socket provided on the sound source board 41 in this arrangement. Additionally, the extension interface 27 may be provided with an additional D/A converter 28. Otherwise, the sound source 22 may be provided in the form of a discrete LSI chip, or mounted on the daughter board. The chip or daughter board is installed through a socket provided on the mother board or extension board in this arrangement. Similarly, the DSP 21 may be provided in the form of a DSP board 42. In this case, the DSP 21 communicates with the CPU 10 via the extension interface 27. Otherwise, the DSP 21 may be provided in the form of a discrete LSI chip likewise the sound source 22. The input data is transferred to the D/A converter 23 through the bus 12 in Figure 1. However, the data can be distributed directly or through the extension interface, if the DSP 21 or sound source 22 is installed through the socket or the extension interface.

[0017] As shown in Figure 3, a first waveform generator or sound source system 32 comprised of the sound source 22 or the DSP 21 can be connected to a local bus 33, through which the data is transferred to and from a CPU system 30 without using the data bus 12. The CPU system 30 is composed of a standard arrangement including the CPU 10, ROM 11 and RAM 13, while peripheral devices 31 include multi type I/O port 14, storage unit 15, and miscellaneous interfaces and operators. The sound source system 32 is specifically composed of the discrete sound source 22 or DSP 21 in the embodiment. However, in general, any functional elements for the musical sound generation are involved in the sound source system 32. The sound source system 32 may be integrated with or separated from the CPU system 30. Further, the connection interface can be provided at either of the CPU side and the sound source device side. Namely, any connection interface may be employed according to the inventive system setup. Additionally to the local bus, any sort of interface/protocol combination such as MIDI, RS-232C/422, IEEE P-1394, or SCSI may be employed. Also, communication network such as public telephone network may be used as the data communication media.

[0018] In an arrangement shown in Figure 4A, the sound source device 22 may be integrated into a single chip, or into a printed circuit board module together with the waveform memory 25 and the D/A converter 23. Similarly, as shown in Figure 4B, the DSP 21, RAM 24, and D/A converter 23 may be integrated altogether into a single chip.

[0019] The arrangements shown in Figures 1 to 3 are nothing but an example. The style of the device connection depends upon the individual system setup. Further, two or more of the functional elements shown in Figure 1 may be integrated into a single chip.

[0020] According to the invention, the above constructed musical sound generating apparatus creates a waveform to generate a musical sound according to performance information. A first waveform generator such as the sound source 22 or the DSP 21 is operable for creating a waveform. A second waveform generator composed of the CPU 10 is operable independently from the first waveform generator for creating a waveform. The I/O 14 provides performance information. One of the first waveform generator and the second waveform generator is designated in correspondence with the provided performance information. The apparatus selectively operates the designated one of the first waveform generator and the second waveform generator to create the waveform according to the provided performance information. The DAC 23 generates the musical sound based on the created waveform. The first waveform generator comprises an external waveform generator optionally connectable to the apparatus while the second waveform generator comprises an internal

waveform generator integrated with the CPU 10. Occasionally, the internal waveform generator is designated in place of the external waveform generator when the same is not connected to the apparatus even though the external waveform generator should primarily correspond to the provided performance information. The second waveform generator is integrated with the CPU 10 to constitute a main part composed of a computer, while the first waveform generator alone constitutes a supplementary part which is separate from the main part and which is optionally installable in the computer. The first waveform generator is composed of a hardware module driven by the CPU 10, while the second waveform generator is composed of a software module installable in the computer.

[0021] Various operating modes in the present embodiment will be described hereunder. The operating modes of the inventive musical sound generator can be categorized, as shown in Figure 5, into two major groups, one of which relates to designation of a synthesizing method, and the other of which relates to allocation of a timbre to the different waveform generators. The two major groups are divided into more specific modes. First of all, the modes to specify the synthesizing method will be described hereunder.

[0022] In the present embodiment, the sound generation is carried out by synthesizing a waveform or wave data of a musical sound according to the performance information, and by converting it into an analog signal. The wave data can be generated in various methods. The used method is determined according to the operating mode in which the synthesizing method is specified. In the present embodiment, CPU synthesizing mode by the second waveform generator and sound source synthesizing mode by the first waveform generator are assumed by selection.

[0023] In the CPU synthesizing mode, a musical sound is synthesized only by the CPU 10, or by combination of the CPU 10 and the co-processor 17. Further, the CPU synthesizing mode can roughly be divided into the following four submodes. The generated waveform is converted into an analog signal by the D/A converter 23 for acoustic reproduction.

FM mode: This FM mode utilizes a software module of an FM sound source for synthesizing a sound. The wave data is generated by real-time FM modulation over a basic sine wave by means of the CPU 10.

Harmonics synthesizing mode: The harmonics synthesizing mode is such that a fundamental waveform and its harmonics are synthesized altogether. With the real-time operation by the CPU 10, a fundamental waveform and its harmonics are calculated to synthesize the waveform.

Wave form memory readout mode: In this mode, the sound is synthesized by accessing a waveform memory. Prior to the synthesizing, the CPU 10 loads a plurality of basic waveforms into the RAM 20. Upon a synthesizing command entry, the CPU generates the wave data of a specified timbre at a specified pitch and volume by reading out the waveform. In the waveform memory readout mode, it is possible to synthesize a sound even with a low performance CPU, since the synthesizing is carried out by accessing RAM or ROM to read out wave data. Thus, a work load of the CPU in this mode is smaller than that in the FM mode and the harmonics synthesizing mode. However, the RAM should be allocated with a wave data area, so that shortage of a free area of the RAM 13 or 20 may be caused occasionally. Thus, the waveform memory readout mode may not be used preferably under some situations dependently on total RAM capacity and CPU addressing volume.

Physical model synthesizing mode: In the physical model synthesizing mode, the sound generation mechanism of an actual musical instrument, such as air flow in a tube, is simulated by an electronic model in order to synthesize the sound. The wave data is computed with real-time operation of devices including the CPU 10. An example of the algorithm for the physical model synthesizing is disclosed in JP-A-63-40199.

[0024] As listed above, the CPU-aided software sound source or the second waveform generator includes a plurality of digital waveform generators which are operable based on different algorithms to arithmetically create digital waveforms having different qualities. The inventive apparatus selectively operates an optimal one of the digital waveform generators according to the provided performance information. Specifically, the second waveform generator includes a digital waveform generator of the waveform memory readout type operable based on a relatively simple algorithm to create a digital waveform having a relatively low quality, and other digital waveform generators operable based on a relatively complicated algorithm to create another digital waveform having a relatively high quality.

[0025] On the other hand, in the hardware sound source synthesizing mode, the musical sound is synthesized using a specific hardware such as the LSI sound source 22. As a matter of course, in this mode, the hardware module such as the LSI sound source 22 must be installed in the system. The LSI sound source 22 synthesizes the wave data by the FM mode or the waveform memory readout mode (likewise the software sound source). The synthesizing method is determined by the hardware itself. The CPU 10 does not cover the control of the native synthesizing process of the sound source 22.

[0026] In the present embodiment, a multiple of voice

channels are provided. One channel is allocated for one tone in either of the CPU synthesizing mode and the sound source synthesizing mode. A plurality of musical sounds are generated in the multiple of the channels to realize concurrent sounding of the plural voices. Since the wave data can be synthesized by both of the CPU 10 and the sound source device 22 in the present embodiment, selection of the waveform generators to be utilized is an important issue. In the present embodiment, optimum one of the waveform generators is designated according to the voice allocation upon accepting a note-on command. The allocation mode includes the followings:

CPU select mode: In the CPU select mode, the waveform synthesizing is effected by the CPU synthesizing mode at first priority. However, if the capability of the computing devices including the CPU 10 is not enough, the number of voice channels which can be used for the synthesizing is limited. In such a case, a part of the waveform synthesizing operation exceeding the capacity of the computing devices including the CPU 10 is carried out by the hardware sound source.

Sound source select mode: In the sound source select mode, the waveform synthesizing is effected by the hardware sound source at first priority. However, if the capability of the hardware sound source device 22 is not enough, the number of the channels which can be used for the synthesizing is limited. In such a case, a part of the waveform synthesizing operation exceeding the capacity of the sound source device 22 is carried out by the CPU software sound source.

Manual mode: In the manual mode, the user manually specifies either of the software and the hardware sound sources. Further, a specific synthesizing mode is designated if the CPU-aided software sound source is specified.

Compulsory mode: In the compulsory mode, the sound source to be used is forcibly determined according to running state of application programs other than the sound generation program without regard to the user's intention.

[0027] A memory map of the RAM 13 or 20 will be described hereunder. The arrangement of the inventive musical sound generator is not so different from general personal computers. Further, the general personal computer can be used as the musical sound generator provided that it executes operations related to the waveform synthesis. Thus, the content of the RAM 13 or 20 is not so different from that of the personal computer. The memory space of the RAM 13 or 20 is divided into a plurality of areas as shown in Figure 6. In Figure 6, an OS area is occupied by the operating system as in the general personal computer. The application program areas (1) to (n) accommodate various application programs other than the waveform synthesizing program. These areas are allocated one by one for the invoked application programs. Song data and other miscellaneous data are stored in a data area, while the wave data is loaded into a wave data area WAVE when the synthesizing is carried out by the waveform memory readout method. Lastly, a designated one of waveform synthesizing programs is stored in a waveform synthesizing program area.

[0028] The operation of the musical sound generator according to the invention will be described hereunder. The musical sound is generated by executing a specific application program of the personal computer, namely, the waveform synthesizing program. Otherwise, the waveform synthesizing program may be implemented as a facility of the OS in the form of a transient program which is automatically installed at the time of boot-up of the system. Even though the memory address and execution permission of the waveform synthesizing program depends on configuration of the OS environment, user operation, number of application programs, operating conditions etc., the waveform synthesizing program is executed as one of the applications (1) to (n). Upon power-on or reset of the musical sound generator, as shown in Figure 7, various registers and flags are set/reset for initialization in step S1. In step S2, system administration process of the OS is executed. In steps S3 to S5, the application program (1), the waveform synthesizing program, and the application program (n) are respectively executed. The waveform synthesizing program is executed to generate one sample value of the wave data at one cycle of the program invocation. The application programs (1) to (n) do not include the waveform synthesizing program. These application programs may be related to music performance, or to entirely different affairs. After step S5, the procedure returns to S2.

[0029] If there is no change in execution status of the application programs, the loop of S2 to S5 is repeatedly executed. Otherwise, if there is a change in the execution status of the application programs, such a change is detected at the system administration process in step S2. If the change of the status is of program termination, the relevant execution step of the application program is skipped. If the change of the status is of program invocation, a step to execute a new application program is added in the loop, and the whole loop is executed repeatedly. Thus, the executing period of the loop changes dependently on the running situation of the application programs and the system load. However, regardless of the running situation of the application programs, one sample value of the wave data of the musical sound is always generated per loop. A series of the sample values are generated continuously by repeating the loop to create a desired waveform. Thus,

if the generated wave data is simply converted into an analog signal, the sampling period is varied, so that jitter may occur in the reproduced musical sound. The data buffer is provided before the D/A converter 23 in order to temporarily store the generated wave data of the sound. The data buffer is accessed for readout of the wave data at a fixed sampling frequency fs. If the musical sound generation is conducted by a fixed program in a certain case where the system is not a personal computer but a stand-alone electronic musical instrument, sound source module, or any other systems having a facility to generate sound, the execution period of the loop process can be fixed. In other words, the loop process is executed at a fixed interval. In such a case, it is very practical to make the loop interval to coincide with the reciprocal of the sampling frequency fs so that the data buffer can be eliminated.

[0030] The waveform synthesizing program executed in step S4 is described hereunder referring to Figures 8 to 11. The program is executed after loading thereof from the storage unit 15 according to a predetermined operation. In step Sa1, the synthesizing mode and the hardware setup are checked. In the hardware setup check, optional devices are recognized by the check method described before. As for the operating mode, both of the synthesizing mode and the voice allocation mode are checked as well. With respect to the operating mode setup, if other application programs are executed before invoking the waveform synthesizing program, the voice allocation mode may be turned to the compulsory mode. Otherwise, the synthesizing mode and the voice allocation mode may be set up according to the user choice via a displayed menu to input desired settings. Further, if various sound source devices are recognized in the hardware check, it is possible to set either of the CPU select mode or the sound source select mode. Thus, in the operating mode setup in step Sa1, the operating mode is set up and recognized before executing the waveform synthesizing program.

[0031] In step Sa2, waveform loading process is executed. In the waveform loading process, typical or basic waveforms are loaded into the area WAVE allocated in the RAM 13 or 20 in case that the basic waveform is used in the waveform memory readout mode. In step Sa3, it is tested whether a flag SETFLG is "1" or not. The flag SETFLG is initially set to "0", but may be turned to "1" if the sampling frequency fs is set up in step Sa21, or if the waveform memory readout mode is designated in a backup waveform computation mode. The procedure advances forward to step Sa4 if the flag SETFLG is "1". Otherwise, if the flag SETFLG is "0", the procedure jumps to step Sa5. In case that the waveform synthesizing program is executed in the loop shown in Figure 7 at the first time, the flag SETFLG is "0", and the procedure unconditionally advances forward to step Sa5. However, the process of step Sa4 is also described here, for convenience of description. In step Sa4, it is tested whether the sound synthesizing should be carried out entirely by the CPU synthesizing mode or not. It is possible to use the result of the hardware check in step Sa1 to detect whether any external sound source device is recognized or not. If there is possibility of using any sound source other than CPU for the waveform synthesizing, step Sa4 branches to "No" direction. In step Sa5, a flag ENBLFLG is tested if it is "1" or not, in order to detect a nonoperable state. The nonoperable state means that the sampling frequency fs is neither set up in the waveform generation procedure, nor the backup waveform computation mode is enabled. Therefore, the CPU synthesizing mode is not yet ready in the nonoperable state.

[0032] When the waveform synthesizing program is executed in the main loop shown in Figure 7 at the first time, the flag ENBLFLG is "0" so that the procedure advances forward to step Sa11. However, the process under an operable state in case that the flag ENBLFLG is "1" is described here just for convenience of description. The operable state means that all the preparation for the CPU synthesizing is already completed. In this state, the procedure advances forward to step Sa6, where processing of performance information is done. In step Sa7, existence of a CPU waveform generation command is checked. The CPU waveform generation command is generated in response to a key-on event contained in the keyboard information KBD, MIDI information or performance information fed from the I/F under the CPU synthesizing mode. If the CPU waveform generation command is detected in step Sa7, the procedure responsive to the command is executed in step Sa8. In the procedure of step Sa8, the wave data is generated by a specific synthesizing mode selected out of the available CPU synthesizing modes. The wave data is then distributed to the D/A converter 23 via the bus 12. Thus, the sound generation is accomplished according to the synthesized wave data or waveform.

[0033] In broader definition, the CPU waveform generation command may include a key-off event commanding note-off, though explanation for the procedures relevant to the key-off event is omitted here. These note-off procedures are very simple process such as release of the waveform generation and termination of the waveform generation. If no CPU waveform generation command is detected in step Sa7, the waveform synthesizing computation in step Sa8 is skipped, since there is no job to be executed. In step Sa9, it is tested whether terminating operation of the waveform synthesizing program is conducted by the user or not. If the termination of the program is not indicated, the procedure immediately returns to prepare for a next CPU waveform generation command. On the other hand, in case that the termination command is inputted, the waveform synthesizing program is terminated by setting the flag SETFLG to "0" in step Sa10.

[0034] If the waveform synthesizing program is executed at the first time in the main loop of Figure 7, the flag ENBLFLG is "0". Then, in step Sa11, it is tested

whether the allocation mode is either of the CPU select mode and the sound source select mode. If the allocation mode is the manual mode or compulsory mode, the detection result is "No" so that the procedure advances forward to step Sa12, where the flags ENBLFLG, DACENBL, and SETFLG are all set to "1". Then, the procedure returns. The flag DACENBL is set to "1" to enable the D/A converter 23. Thus, if the waveform synthesizing program is executed next time and the external sound source is used, the check of step Sa3 results in "Yes" and the check of step Sa4 results in "No" so that the procedure shown in Figure 11 is executed. Otherwise steps Sa6 to Sa10 are executed if the waveform synthesizing is effected by only the CPU.

[0035] On the other hand, when it is detected that the allocation mode is either of the CPU or sound source select mode in step Sa11, the procedure advances forward to step Sa13. The processing in steps Sa13 to Sa27 shown in Figures 9 and 10 may be done if the waveform generation program 15 invoked at the first time. In this process, the sampling frequency fs is determined for synthesis only by the CPU. In step Sa13, the flag ENBLFLG is switched to "0", a flag BUSY is set to "1" and the flag DACENBL is set to "0" explicitly. the flag BUSY is set to "1" to enable counting up in timer process described later. Setting of the flag DACENBL to "0" is done in order to disable the output operation of the D/A converter 23 to prevent sound generation during the trial waveform computation described later. After step Sa13, registers SCOUNT and TCOUNT are reset to "0". The content of the register SCOUNT indicates loop cycles of the following trial waveform computation. The register TCOUNT incrementally counts up while the flag BUSY is switched to "1". Thus, this register indicates a lapse time required to generate one waveform by m cycles of the synthesis computation. In step Sa15, one sample value for one sampling period is generated by the execution of a predetermined wave data generation algorithm. The detail of the processing in step Sa15 will be described later. In step Sa16, the register SCOUNT is incremented by "1" whenever the sample value computation is executed once. In step Sa17, it is tested if the register SCOUNT reaches cycle "m". If it is detected NO, the procedure returns to step Sa15. Otherwise, the procedure advances forward to step Sa18 in Figure 10 in case that the check result is YES. Thus, the steps Sa15 and Sa16 are repeated until the loop cycle of the sample value computation reaches to "m".

[0036] In step Sa18 of Figure 10, the flag BUSY is switched to "0" to disable the counting up of the timer. Then, in step Sa19, a frequency Fs is calculated by the following equation (1).

$$Fs = (m \cdot margin)/(TCOUNT \cdot Tt) \quad (1)$$

In this equation, the "margin" is a constant set smaller than value 1 to provide the computation devices including the CPU 10 with some margin, considering the processing power of the devices. As described before, TCOUNT indicates invocation times of the timer during the period required to "m" cycles of the execution of the sample value calculation, while Tt indicates a pitch of the timer. Thus, the product of TCOUNT and Tt corresponds to the period required for completing the sample value calculation process to generate one waveform.

[0037] Thus, the frequency Fs calculated by the equation (1) is the frequency of the waveform sampling, and the constant "margin" reflects the processing power of the hardware setup.

[0038] In step Sa20, it is detected whether the calculated frequency Fs is greater than 32 kHz or not. This critical frequency "32 kHz" is employed considering the minimum quality of the musical sound to be generated. If this detection result is positive, which means that the processing power of the CPU is evaluated as sufficient for maintaining the minimum quality of the sound, the procedure advances forward to step Sa21. In step Sa21, an adequate sampling frequency fs smaller than and closest to the calculated frequency Fs is selected out of 32 kHz, 44.1 kHz, 48 kHz, and 50 kHz. If the calculated frequency Fs is 47 kHz, the sampling frequency fs is set to 44.1 kHz, which is closest to and smaller than 47 kHz. The value smaller than the calculated frequency Fs is selected because the constant margin does not make sense if the sampling frequency fs is set to exceed the processing power of the CPU. Alter step Sa21, the flags DACENBL, ENBLFLG and SETFLAG are all set to "1" in step Sa22, and the procedure returns. This flag operation enables the D/A converter 23 to output the musical sound. Further, this flag operation indicates that the sampling frequency fs is set up. Consequently, the nonoperable state is changed to the operable state.

[0039] Under the operable state, upon the next execution of the waveform synthesizing program, check of step Sa4 results in "No" so that the synthesizing is done by the processing shown in Figure 11 if the waveform synthesizing should be carried out by the external sound source device. Otherwise, if only the CPU is used for the waveform synthesizing, check of step Sa4 results in "Yes" and check of step Sa5 results in "No" so that the synthesizing is carried out through steps Sa6 to Sa8.

[0040] On the other hand, in Figure 10, if the result of the test about the calculated frequency Fs in step Sa20 is turned out "No", which means that the minimum quality of the sound cannot be retained, the procedure advances forward to step Sa23. In the step Sa23, the user is warned that the minimum quality of the sound is not available, and the backup waveform computation mode is called. The backup waveform computation mode is the waveform memory readout mode which is selected as a secondary choice when the minimum quality of the sound is not affordable with the primary synthesizing mode selected out of the available CPU synthesizing modes. In step Sa24, it is tested whether the backup waveform computation mode is designated

or not. If YES, the procedure goes forward to step Sa25, where original waveforms are prepared by the primary computation mode and are stored in the RAM 13 or 20 in advance. Further, the sound is actually reproduced when the stored waveform is read out at the sampling frequency 32 kHz, which is automatically determined. Thus, the waveform synthesizing is effected virtually with the waveform memory readout mode, so that the minimum quality of the reproduced sound can be retained even if the low performance CPU is equipped in the system. After that, the process in step Sa22 is executed to switch the nonoperable state to the operable state, and the procedure returns. On the other hand, if the backup waveform calculating mode is not specified in step Sa24, the procedure goes forward to step Sa26, in which a termination command for the sound synthesizing program is detected. If the termination of the sound synthesizing program is commanded, the operation finishes, by clearing the flag SETFLG to "0" in step Sa27. In case that the calculated frequency Fs is smaller than 32 kHz, and the backup waveform calculation mode is neither specified nor the program termination is commanded, the warning alarm is continued. Thus, the waveform synthesizing is done with the computation devices such as the CPU 10. The sampling frequency fs is optimized with respect to the CPU performance. Further, if the performance of the CPU is low, the synthesizing is executed by switching to the waveform memory readout mode, which can reduce the CPU load.

[0041] For summary, the CPU-aided second waveform generator comprises a computerized waveform generator operable according to a given algorithm at a variable operation speed to create a digital waveform by successively computing sample values of the digital waveform. The computerized waveform generator is provisionally operated to carry out trial creation of a model digital waveform while measuring the operation speed at which the trial creation is carried out. The sampling frequency is optimally determined in comparable to the measured operation speed. The computerized waveform generator is actually operated to enable the same to successively compute sample values of an actual digital waveform at the determined sampling frequency according to the provided performance information. The sample frequency is fixed to one of stepwise predetermined levels, which is lower than and closest to the measured operation speed. When the determined sampling frequency falls below a critical level which is defined to ensure a minimal quality of the digital waveform, the initial algorithm is changed to raise the operation speed of the computerized waveform generator so that the sampling frequency can be redetermined to exceed the critical level. The algorithm is changed from a complicated one to a simplified one such that the computerized waveform generator operates based on the simplified one of the algorithm to successively read out prestored ones of sample values to reproductively create the digital waveform. The sample values may be initially computed by the complicated algorithm. Then, the sample values are stored in the waveform memory for actual use under the simple algorithm of the waveform memory readout mode. Thus, the inventive sound generating apparatus for creating a digital waveform to generate a musical sound according to performance information, comprises input means for providing performance information, computerized waveform generator means operable based on a given algorithm at a variable operation speed to create a digital waveform by successively computing sample values of the digital waveform, trial means for provisionally operating the computerized waveform generator to carry out trial creation of a model digital waveform while measuring the operation speed at which the trial creation is carried out, determining means for optimally determining a sampling frequency comparable to the measured operation speed, controller means for actually operating the computerized waveform generator to enable the same to successively compute sample values of an actual digital waveform at the determined sampling frequency, and output means for generating the musical sound based on the actual digital waveform according to the provided performance information.

[0042] By the way, if there is detected a possible use of the hardware sound source device for the waveform synthesizing in step Sa4 in Figure 8, the procedure branches to step Sa28 in Figure 11. In step Sa28, the flag DACENBL is switched to "1" to enable the D/A converter 23 to output the sound. In step Sa29, any termination command for the sound synthesizing program is detected. If the termination of the sound synthesizing program is commanded, the synthesizing program finishes by clearing the flag SETFLG to "0" in step Sa31 through step Sa30. On the other hand, if the termination of the sound synthesizing program is not commanded, the operating mode is checked whether it is the sound source synthesizing mode or not in step Sa32, if the detection result of step Sa29 is negative. When the sound source synthesizing mode is detected in step Sa32, existence of the hardware required for the specified mode is detected in step Sa33. This mode is specified upon the recognition with the hardware check in step Sa1. If the relevant hardware exists, the synthesizing procedure is executed by the relevant hardware in step Sa34. Otherwise, the lack of the relevant hardware is issued in step Sa35, and the synthesizing process is continued using the current hardware setup, if the relevant hardware does not exist. After steps Sa34 and Sa35, the procedure returns. The synthesizing processing with the selected hardware will be described later.

[0043] The waveform loading process executed in step Sa2 (Figure 8) will be described hereunder with reference to Figure 12. In the waveform loading process, it is detected whether a MIDI sample dump command is received through the multi I/O port 14 in step Sb1. The MIDI sample dump contains a wave data

according to the MIDI standard, and is used in the waveform memory readout mode. If the MIDI sample dump is received, the sample receiving procedure is conducted such that the received data is transferred to the area WAVE in RAM 13 or 20 in step Sb2. The sample dump receiving of step Sb2 is repeated until all of the data reception are completed. The completion is detected in step Sb3. If all of the data reception are completed, the detection in step Sb3 results in "Yes", and the procedure returns. On the other hand, if the check of step Sb1 results in "No", it is checked if the wave data is transferred via the interface I/F in step Sb4. If the wave data is received, the received data is transferred to the area WAVE in the RAM 13 or 20 in steps Sb2 and Sb3, as in the case of the MIDI sample dump. If both the steps Sb1 and Sb4 result in "No", an access reading event for the storage unit 15, namely the request to read out some data from the storage unit 15, is detected in step Sb5. If the request is not issued, the waveform loading is terminated immediately and the procedure returns, since there is no more job to do. On the other hand, when the access event occurs, the data read out from the storage unit 15 is checked in step Sb6. Further, in step Sb7, it is tested whether the read out data is the wave data or not. If the read out data is not the wave data, the waveform loading is terminated immediately and the procedure returns, since there is no more job to do. Otherwise, if the wave data is read out, the wave data is transferred to the area WAVE in the RAM 13 or 20. The data transfer in step Sb8 is repeated until the end of the data transfer is detected in step Sb9. When the data transfer is completed, the check of step Sb9 results in "Yes" and the procedure returns. Thus, in the waveform loading, the wave data to be used in the waveform memory readout mode is received or read out. Then, the wave data is transferred to the area WAVE in the RAM 13 or 20. The computation devices such as CPU 10 process the wave data with the waveform memory readout mode according to a simple algorithm in order to synthesize the actual waveform. In order to reproduce the wave data loaded in the RAM 13 or 20 by the sound source 22, the loaded basic wave data might be transferred again to the waveform memory 25 included in the sound source 22. In a conventional musical sound generator having a specific hardware sound source, the hardware sound source must have a temporary waveform memory to receive the loaded wave data. The CPU must execute the wave data transfer from the RAM to the waveform memory. However, in the present embodiment, the basic wave data is loaded to RAM 13 or 20 under the control of the CPU 10. Therefore, it is not necessary to provide the temporary memory to hold the wave data in the hardware. Further, it is not necessary to execute re-loading process in which the loaded wave data is transferred further to the external hardware device. The costs for the system hardware or software can be reduced, and the time from the end of the wave data loading to the reproduction of the sound can be shortened.

[0044] The sample value computing operation executed in step Sa15 (Figure 9) will be explained hereunder with reference to Figure 13. In step Sc1 of Figure 13, it is tested whether the co-processor 17 exists. In the present embodiment, the co-processor 17 is installed. However, this device is optional and may be lacked in some hardware setup. If the CPU 10 accommodates an arithmetic operation unit equivalent to the co-processor 17, the detection of the co-processor is not required. It is possible to regard the system as if the co-processor is installed. If the co-processor 17 is detected, the sample value calculation is done with the CPU 10 as well as the co-processor 17 in step Sc2. Otherwise, the sample value calculation is carried out only with the CPU 10 in step Sc3, if there is no co-processor 17. After step Sc2 or Sc3, the procedure returns. The sample value calculation is the same in steps Sc2 and Sc3, except that the co-processor 17 is used or not. Thus, the detail of the sample value calculation is explained with respect to step Sc3 here.

[0045] In step Sd1 of Figure 14, it is tested whether the current operating mode is set to the FM mode among the various CPU synthesizing modes. If the FM mode is set, a set of sample values corresponding to a number of channels are calculated at one sampling point according to the FM synthesizing method in step Sd2. For example, if polyphonic synthesizing is selected, a set of multiple sample values required for synthesizing a desired number of voices are calculated. In this case, the load of the CPU 10 is high, since various voices having various pitches may be concurrently created. Even worse, other processing such as graphics processing may be done in parallel. If the FM mode is not detected in step Sd1, it is detected whether the current operating mode is set to the waveform memory readout mode among the CPU synthesizing modes in step Sd3. If the waveform memory readout mode is detected here, a set of sample values required for one sampling point are read out from the waveform memory. In polyphonic synthesizing, multiple samples required for synthesizing a desired number of voices are read out. This data reading and data transfer is not carried out by the CPU 10, but by the DMAC 19. The wave data has been loaded to the area WAVE in the RAM 13 or 20, or provisionally stored in the ROM 11. If the waveform memory readout mode is not detected in step Sd3, it is detected whether the current operating mode is set to the harmonics synthesizing mode among the various CPU synthesizing modes in step Sd5. If the harmonics synthesizing mode is detected, a set of sample values required for one sampling process are calculated by the CPU 10 according to the harmonics synthesizing method in step Sd6. In polyphonic mode, a number of samples required for synthesizing a desired number of voices are calculated with the harmonics synthesizing method. In this case, the load of the CPU 10 is high, since various voices having various pitches may be cre-

ated concurrently. Even worse, other processings may be carried out in parallel, similarly to the situation under the FM mode. If the harmonics synthesizing method is not detected in step Sd5, it is tested whether the current operating mode is set to the physical model synthesizing mode among the various CPU synthesizing modes in step Sd7. If the physical model synthesizing mode is detected, samples required for one sampling point are calculated by the CPU 10 according to the physical model synthesizing method in step Sd8. In polyphonic mode, a number of samples required for synthesizing a desired number of voices are calculated with the physical model synthesizing method. If the physical model synthesizing mode is not detected in step Sd7, the current operating mode is out of the present embodiment, so that the warning for the wrong mode setting is executed in step Sd9 as well as other process. After steps Sd2, Sd4, Sd6, Sd8, or Sd9, the procedure returns, and then the following step Sa16 (Figure 9) is executed.

[0046] As for the waveform sample value calculation with the CPU 10 and the co-processor 17 executed in the step Sc2, the procedure shown in Figure 14 is executed with cooperation of the CPU 10 and the co-processor 17 so that the calculating operation can be made faster. The detailed explanation for the operation is omitted here, since the procedure is basically the same as the calculation only by the CPU.

[0047] Thus, in the waveform sample value calculation in step Sa15, the most influential processing on the quality of the synthesized sound is carried out by one of the CPU synthesizing modes. In the FM, harmonics synthesizing, or physical model synthesizing mode, the time required for the wave data calculation, as well as the number of voices to be reproduced simultaneously, is the bottle neck of the sound synthesizing. The wave data at each sample point is actually calculated in order to examine the processing power. The calculation is repeatedly done in steps Sa15 to Sa17 for m cycles, and the time required for the m sample calculation is measured by the timer operation to set the sampling frequency fs suitable for the processing power of the CPU in step Sa21. Similarly in the waveform memory readout mode, the readout operation of the wave data is the bottle neck of the sound synthesizing, so that the wave data at each sample point is read out to examine the accessing speed. The readout process is done in steps Sa15 to Sa17 for m samples, and the time required for the m sample calculation is measured by the timer operation to set the sampling frequency fs suitable for the processing power of the CPU in step Sa21.

[0048] The synthesizing by the selected hardware, which is executed in the above-described step Sa34 (Figure 11), is explained hereunder. This synthesizing process is effected unless voices of all the channels are synthesized by the CPU synthesizing mode alone. This process is executed to control the sound source device used for synthesizing the waveform according to the allocation mode. First of all in this procedure, event detection is carried out in step Se1 of Figure 15. The event includes a key-on event issued in response to keyboard information KBD or MIDI information, and includes other events not only associated to the CPU synthesizing mode, but also associated to the sound source device mode. Upon detecting the key-on event, the apparatus initiates the synthesizing process. In step Se2, it is tested whether the current operating mode designates the sound source synthesizing mode or not. If the detection result is "No", the procedure branches to step Se11. Otherwise, the procedure goes forward to following step Se3, if the detection result is "Yes". In step Se3, it is tested whether the current synthesizing operation status of the sound source devices and the performance information corresponding to the event comply with a "condition" for synthesis of waveform by the sound source device. In the present embodiment, one condition is whether the number of voices (timbres) to be synthesized currently is within a maximum number of voices which can be synthesized simultaneously by the hardware device specified in the sound source synthesizing mode. More particularly, in step Se3, the number of channels currently being in active status and being allocated for the sound source device is less than or equal to the full number of the channels which can be used for simultaneous synthesis of voices by the device. The "condition" may include other factors listed hereunder: (1) Whether a "pitch" or a "touch" specified by the detected event is higher (or lower) than a predetermined value. (2) Whether a value relevant to "timbre" specified by the detected event is higher (or lower) than a predetermined value. (3) Whether a value relevant to a number of "parts" in the performance information specified by the detected event is higher (or lower) than a predetermined value. (4) Whether a detected MIDI-CH value (number of channels) relevant to the detected event is higher (or lower) than a predetermined value. As shown above, the criteria for the "condition" may be generalized as whether a certain value specified by the performance information is higher (or lower) than a predetermined value. It is possible to implement a particular "condition" to issue a negative result if some unique timbre is specified to be synthesized in the FM mode or the harmonics synthesizing mode using the sound source device. This implementation will be described later in another embodiment. If the "condition" is fulfilled, channel assignment process is conducted in step Se4 such that a channel to synthesize a voice for the key-on event is allocated out of vacant channels in the sound source device, which are currently not used for synthesizing the sound. In step Se5, the sound generation is executed using the specified sound source hardware, in which the actual synthesizing of the waveform is done for the issued event in the allocated channel.

[0049] If the "condition" is not fulfilled in step Se3, it is tested whether the flag ENBLFLG is "1" or not in step Se6. In this stage, the value "1" of the flag ENBLFLG indicates that the steps Sa13 to Sa25 have been exe-

cuted already, and indicates that the sampling frequency fs for the CPU synthesizing mode is already set up. Thus, the CPU synthesizing mode is available. Therefore, if the test result of step Se6 is "Yes" because of value "1" of the flag ENBLFLG, voice allocation procedure to the CPU synthesizing mode is executed in step Se7 so that the synthesizing of the waveform relevant to the event which does not comply with the condition for synthesizing by the sound source device is carried out by the CPU synthesizing mode. Particularly, an allocation command is issued to calculate sample values of the waveform relevant to the event in this CPU synthesizing allocation procedure. The command includes designation of computation mode to be executed by the CPU 10 (any mode out of the CPU synthesizing modes), and designation of the timbre, pitch, touch, volume and channel assignment. Further, the command includes note commands such as key-on or key-off. During the allocation command is valid, the computation devices including the CPU 10 execute the waveform synthesizing computation process in order to generate the sample values relevant to the event in step Se10. The allocation command also includes information about the start and end of interrupts, if the waveform is generated with the interrupt process. On the other hand, if the value "0" of flag ENBLFLG is detected in step Se6, the CPU synthesizing mode is not available. In this case, truncating process is carried out to turn off sound reproduction operation of a channel which is the oldest in step Se8, to make a vacant channel forcibly. This truncating process may be included in the allocation process executed in step Se4. In step Se4, the relevant event is allocated to the vacant channel made by the forcible note-off, and the synthesizing of the waveform relevant to the event is carried out by the allocated channel in step Se4. If multiple sound source devices are installed to the system, channels of the different sound source devices may be allocated to synthesize a single timbre.

[0050] By the way, if the sound source device select mode or preceding mode is not detected in step Se2, there may be the CPU select preceding mode or the manual mode in which the CPU synthesizing mode is set together with the sound source synthesizing mode. Thus, the procedure branches to step Se11 in Figure 16. In this step Se11, it is tested whether the current state of the CPU synthesizing mode and the performance information relevant to the issued event comply with the "condition" for synthesizing by the devices including the CPU. Various factors may be conceived as the "condition" for synthesis by CPU as in the case of the "condition" for synthesis by the sound source device (step Se3). In this embodiment, the condition is whether a number of voices (timbres) to be synthesized currently is within a maximum number of voices which can be synthesized simultaneously with the CPU synthesizing mode. More particularly, in step Se11, it is tested whether the number of channels (CH) currently being held in note-on status and being allocated for the computation devices including the CPU is less than or equal to the maximum number of the channels which can be used for simultaneous synthesizing of the waveforms by the CPU synthesizing mode. If the "condition" is satisfied, it is tested whether the flag ENBLFLG is "1" or not in step Se12. As described above, the value "1" of the flag ENBLFLG here indicates that the synthesis of the waveform by the CPU synthesizing mode is ready to be started. Therefore, the allocation procedure is executed for the waveform synthesis by the devices including the CPU 10 in step Se13, and the procedure further goes forward to step Se9. The detail of the allocation procedure in step Se13 is not explained here again, since the procedure is eventually the same as in step Se7. On the other hand, if the "condition" for synthesis of the waveform by CPU is not complied in step Se11, or the devices including the CPU 10 are not ready in step Se12, the allocation procedure is executed in step Se14 as in step Se4, so that the processing relevant to the event is done by the external sound source device. The synthesizing using the specified hardware is done in step Se15 as in step Se5. After step Se13, steps Se9 and step Se10 are executed. In step Se9, it is detected whether the allocation command exists to generate wave data by the CPU. If the command is not detected, the procedure returns immediately. Otherwise, the waveform is calculated according to the allocation command to generate sound by the CPU in step Se10.

[0051] Thus, in the external sound source select preceding mode, the synthesizing process complying with the "condition" for synthesis by the hardware device is executed by the sound source in step Se5, while the synthesizing process which does not comply with the "condition" is executed by the CPU 10 in step Se10. On the other hand, if the designated mode is not the sound source preceding select mode, the process complying with the "condition" for synthesis by the CPU 10 is executed by the software module in step Se10, while the synthesizing process which does not comply with the "condition" for synthesis by the CPU is executed by the hardware sound source device in step Se15. Thus, in the synthesis using the selected hardware, if the synthesizing process exceeds the processing power of the hardware specified in the sound source synthesizing mode, the exceeding part of the process is executed by the CPU synthesizing mode, so that it is possible to synthesize a plurality of voices more than the full number of voices which can be simultaneously reproduced in the hardware without enhancing hardware setup. In general, when the input device such as the I/O 14 provides performance information effective to command concurrent generation of a plurality of musical sounds, the inventive apparatus designates one of the first waveform generator and the second waveform generator according to a number of concurrent musical sounds specified by the performance information such that the designated one has a capacity sufficient to create a

number of waveforms corresponding to the number of the musical sounds. When the number of the concurrent musical sounds exceeds a capacity of either of the first waveform generator and the second waveform generator, both of the first waveform generator and the second waveform generator are designated to ensure complete generation of the concurrent musical sounds.

[0052] The timer process will be described hereunder. The timer process is an interrupt process executed at a predetermined interval Tt during the trial waveform synthesis program described before. Figure 17 is a flowchart illustrating the timer process in detail. In step Sf1, it is tested if the flag BUSY is "1", which means that the counting up of the timer is enabled. If this detection results in negative, the procedure skips to Sf3. Otherwise, the routine goes forward to step Sf2, where the register TCOUNT is incremented by "1" if the detection result is positive. In step Sf3, other miscellaneous timer processes are executed to finish the timer routine. Thus, the timer process increments the register TCOUNT by "1" if the flag BUSY is "1". The flag BUSY switches to "1" only in the loop of steps Sa9 to Sa11 so that the content of the register TCOUNT indicates the invocation cycles of the timer process within the time required for the computation or reading of one waveform. The lapse time can be derived by multiplying TCOUNT by Tt.

[0053] In the arrangement shown in Figure 1, all the optional devices such as the co-processor 17, DSP 21 and external sound source 22 are fully installed. Thus, in this arrangement, all the operating modes can be designated with respect to the synthesizing mode and the allocation mode. If the operating mode is set to the sound source select mode at first priority, a part of the process exceeding the processing power of the external sound source is allocated to the CPU, and the CPU handles the process so that the musical sound can be generated beyond the limitation of the external sound source device, and various timbres can be generated. Further, various timbres can be generated also in the CPU select mode given first priority. Since the sampling frequency fs is always set to the optimum value at step Sa21, the quality of the generated sound can be maintained high.

[0054] In the embodiment described above, the optional devices are fully installed as shown in Figure 1. However, the hardware setup of the personal computer or the electronic musical instrument varies depending on which optional device is installed. The available operating mode of the musical sound synthesizing program is different depending on the hardware setup. The operations of the musical sound synthesizing program in other hardware arrangements which are different from that in Figure 1 will be described hereunder.

[0055] The hardware setup shown in Figure 21 lacks all the optional devices including the co-processor 17, DSP 21, and LSI sound source 22. Of course, the sound source synthesis mode is not available in this setup, because the detection in step Sa4 (Figure 8) becomes "Yes", and only the CPU synthesizing mode can be utilized. However, if the processing power of the CPU 10 is not very high enough, the real-time wave data computation is impossible, since both of the co-processor 17 and DSP 21 are not installed. Thus, there may be a situation in which the waveform memory readout mode is only available in the CPU synthesizing modes.

[0056] The hardware setup shown in Figure 22 has only the co-processor 17 as an optional device. In this setup, the hardware sound source synthesis mode is not affordable, because the detection in step Sa4 (Figure 8) becomes "Yes". The CPU synthesizing mode is only available. However, all the CPU synthesizing modes are possible, because high speed real time arithmetic operations are possible by the co-processor 17. The actual waveform calculation (step Sa9), as well as the trial waveform calculation (step Sa15), is executed by both of the CPU 10 and the co-processor 17 in this setup, since the co-processor 17 is available.

[0057] The hardware setup shown in Figure 23 includes the co-processor 17 and the DSP 21 as optional devices. The purpose of the DSP 21 is to ensure high speed calculation of the wave data. If the DSP 21 is treated as the external sound source, the sound source synthesis mode is available by means of the DSP on the second invocation of the synthesizing program or thereafter, since the detection in step Sa4 (Figure 8) becomes "No". However, the purpose of the DSP 21 is to facilitate data calculation, so that the wave data is not generated by the waveform memory readout method, but is generated by the various pure arithmetic modes such as FM mode, harmonics synthesis mode and physical modeling mode. Further, the CPU synthesizing mode is available depending on the allocation mode. The wave data calculation is possible, but the waveform memory readout mode is not available since the DMAC 19 and RAM 20 are deleted. It is possible to utilize only the FM mode, harmonics synthesizing mode, and physical model synthesizing mode, where the real-time high speed wave data calculation is executed. The actual waveform calculation (step Sa8), as well as the trial waveform calculation (step Sa15), is executed by both of the CPU 10 and the co-processor 17 in this setup.

[0058] The hardware setup shown in Figure 24, includes only the LSI sound source 22 as an optional device. In this setup, the sound source synthesis mode is available on the second invocation of the synthesizing program or thereafter, since the detection in step Sa4 (Figure 8) becomes "No". Further, the CPU synthesizing mode is available depending on the allocation mode. However, there may be a situation in which the waveform memory readout mode cannot be utilized, since DMAC 19 and RAM 20 are missing. Further, if the processing power of the CPU 10 is not very high enough, the real-time wave data calculation is impossible since the co-processor 17 lacks. Thus, there may be a situation in which all the CPU synthesizing modes are

not available.

[0059] A second embodiment will be explained hereunder. Generally, it is difficult to reproduce the realistic sound of the percussive tones such as a rhythm or drum instrument by the wave data computation according to FM mode or harmonics synthesizing method. Thus, if the LSI sound source 22 is installed and this sound source 22 calculates the wave data of the musical sound by the pure calculating method such as FM mode other than the waveform memory readout mode, it is not adequate to reproduce the sound by the LSI sound source 22. Further, it is not necessary to calculate the wave data by the arithmetic calculating mode other than the waveform memory readout mode by the CPU 10, if this sort of the LSI sound source 22 is installed. Further, the CPU 10 should execute jobs other than the waveform synthesizing, so that a system load required for the execution of the waveform synthesizing program should be reduced as much as possible, especially in case that the processing power of the CPU 10 is not high. Thus, in this situation, it is convenient that the CPU 10 generates the percussive wave data unsuitable for the LSI sound source 22 by the waveform memory readout mode, while the LSI sound source 22 generates the wave data for other timbres. Thus, the computation load can be reduced for the CPU 10, and the LSI sound source 22 does not have to synthesize any wave data for which the sound source 22 has poor ability. The quality of the reproduced sound can be maintained high as much. The purpose of the second embodiment is directed to the very point. The second embodiment assumes that the sound source 22 is installed as shown in Figure 1 or 24. With respect to the waveform synthesizing program, the synthesizing process using the selected hardware (Figures 15 and 16) is replaced with the process shown in Figure 18. More particularly, the synthesizing process using the selected hardware in step Sa34 (Figure 11) is substituted for the process shown in Figure 18. The substituted process will be described hereunder, while omitting the explanation of the other processes for avoiding duplicated description.

[0060] In this second embodiment, upon advancing forward to step Sa34, the program runs to execute the synthesizing process using the selected hardware as shown in Figure 18. In step Sg1, the event detection is carried out as in step Se1. In step Sg2, system check for voice allocation is executed. More particularly, a device to handle the synthesis is determined for each voice (timbre) out of the CPU and the LSI sound source. The criteria for this allocation will be described hereunder. Generally, the sound source has unique disposition of available timbres, so that an individual timbre can be specified by a unique timbre code. Thus, it is possible to implement a table containing the list of the timbre codes of the percussive tones in advance, in order to discriminate between the tones to be handled by the CPU 10 in case that the timbre code detected for the relevant event is found in the table, and the other tones to be handled by the LSI sound source 22. However in the present embodiment, the allocation criteria may not be limited to the timbre code. It is possible to setup timbre handling means under the manual mode, in such a manner that a certain timbre is to be handled by the CPU, and another timbre is to be handled by the sound source. Each tone can also be allocated depending upon the number of the channels which can be used for simultaneous synthesizing. Further, under a compulsory mode, each tone can be allocated forcibly by other running programs.

[0061] For the tones allocated to the LSI sound source, vacant channels are created by the voice allocation process in step Sg3 as in step Se4 (Figure 15). In step Sg4, the waveform relevant to the key-on event is synthesized through the vacant channels. The synthesizing method for this operation is not limited to the FM mode, the harmonics synthesizing mode or the physical modeling mode, but it is possible to use, for example, PCM mode to read out the wave data from the waveform memory 25, depending upon the characteristics of the installed sound source 22. On the other hand, for the timbres allocated to the CPU, the CPU synthesizing allocation procedure is done in order to generate an allocation command to generate wave data relevant to the detected event in step Sg5 as in step Se7 (Figure 15). If the allocation command is available, the detection in step Sg6 results in "Yes", and the synthesizing calculation in step Sg7 is executed to generate a waveform relevant to the allocation command. The synthesis of the waveform is effected by the waveform memory readout mode in order to reduce the load for the CPU as described before. On the other hand, when the allocation command is not available, the detection in step Sg6 results in "No", and the procedure returns.

[0062] In this second embodiment, musical sounds can be allocated selectively to the CPU and the sound source according to their timbres so that the optimum distribution of the processing load for the CPU and the sound source is possible, and the various timbres can be generated while retaining the quality of the reproduced sounds. For summary, the input device provides performance information which contains timbre information effective to specify a timbre of the musical sound and timing information effective to specify a timing of generation of the musical sound. One of the first waveform generator and the second waveform generator is designated in correspondence with the timbre information so that the output device generates the musical sound having the specified timbre at the specified timing.

[0063] In the first and second embodiments, the synthesizing process for one tone is executed in the channels provided on the CPU or in the other channels provided in the sound source. However, the synthesizing process for a tone can be executed by both of the CPU and the sound source. In this arrangement, it is possible to use the harmonics synthesizing mode in the CPU side, and to use a mode other than the harmonics

synthesizing, e.g., the FM mode in the sound source side, so that variational tones can be generated because the same tone is synthesized by the different calculation modes. The purpose of a third embodiment is directed to the very point. In the third embodiment, the LSI sound source is assumed to be installed in the system as shown in Figures 1 and 24. With respect to the waveform synthesizing program, the synthesizing process using the selected hardware device (Figures 15 and 16) is replaced with the process shown in Figure 19. More particularly, the synthesizing process using the selected hardware in step Sa34 (Figure 11) is substituted for the process shown in Figure 19. The substituted process will be described hereunder, while omitting the explanation of the other processes for avoiding duplicated description.

[0064] In this embodiment, upon advancing forward to step Sa34, the synthesizing program runs to execute the synthesizing process using the selected hardware as shown in Figure 19. In step Sh1, the event detection is carried out as in steps Se1 and Sg1. In step Sh2, system check for voice allocation is executed. More particularly, a device to handle the synthesis is determined for each voice (tone) out of the CPU and the LSI sound source. The criteria for this allocation may be the timbre code, number of channels available for simultaneous synthesizing, or the setup configured by the manual or compulsory mode, as in the second embodiment. For the tones allocated to the LSI sound source, vacant channels are prepared by the voice allocation process in step Sh3 as in step Se4 (Figure 15). In step Sh4, the waveform relevant to the note-on event is synthesized through the vacant channels. The synthesizing method for this operation is not limited to the FM mode or the harmonics synthesizing method, but it is possible to use, for example, the physical modeling mode and the PCM mode to read out the wave data from the waveform memory 25 depending upon the characteristics of the installed sound source 22. On the other hand, for the tones allocated to the CPU, the CPU synthesizing allocation procedure is done in order to generate an allocation command relevant to the detected note-on event in step Sh5 as in step Se7 (Figure 15). Information about the calculation method is included in the allocation command. The voice allocated to both of the CPU and the sound source is processed in steps Sh3 and step Sh4 by the LSI sound source, while the same voice is processed in steps Sh5 by the CPU. These processes are executed in parallel. If the allocation command is available, the detection in step Sh6 results in "Yes", and the synthesizing calculation in step Sh7 is executed to generate a waveform relevant to the allocation command. Unlike the second embodiment, the synthesizing calculation is done by various modes including the FM, harmonics synthesizing, physical modeling and so on. On the other hand, if the allocation command is not available, the detection in step Sh6 results in "No", and the procedure returns.

[0065] In the third embodiment, a voice (tone) can be allocated to both of the CPU and the LSI sound source so that different wave data can be reproduced for the same tone actually. Due to this feature, the third embodiment can also diversify the tones of the system. For summary, both of the first waveform generator and the second waveform generator are coincidently designated so that the controller device operates both of the first waveform generator and the second waveform generator to concurrently create waveforms in parallel manner for a single timbre.

[0066] A fourth embodiment will be explained hereunder. Even though the voice allocation mode is introduced in the embodiments described above, more simple implementation is possible. An event which can be allocated to a certain sound source device is simply allocated to the relevant device upon the event detection, provided that the device is installed in the system. The very implementation is provided in this fourth embodiment. The fourth embodiment assumes that the sound source 22 is installed as shown in Figure 1 or 24, as in the second and third embodiments. With respect to the waveform synthesizing program, the synthesizing process using the selected hardware device (Figures 15 and 16) is replaced with the process shown in Figure 20. More particularly, the synthesizing process using the selected hardware device in step Sa34 (Figure 11) is substituted for the process shown in Figure 20. The substituted process will be described hereunder, while omitting the explanation of the other processes for avoiding duplicated description.

[0067] In this embodiment, upon advancing forward to step Sa34, the synthesizing program runs to execute the synthesizing process using the selected hardware as shown in Figure 20. First of all, the event detection is carried out though this process is not illustrated. In step Si1, voice allocation is executed to create a vacant channel in the LSI sound source. In step Si2, the waveform relevant to the detected event is actually synthesized using the vacant channel. The synthesizing method for this operation is not limited to the FM mode or the harmonics synthesizing mode, but it is possible to use, for example, the physical modeling mode and the PCM mode to read out the wave data from the waveform memory 25 depending upon the characteristics of the installed sound source 22. After the synthesizing, the procedure returns. Thus, in the fourth embodiment, all event which can be allocated to a certain sound source device is simply allocated to this device upon the event detection provided that this device is installed in the system.

[0068] Figure 25 shows an additional embodiment of the inventive musical sound generating apparatus. This embodiment has basically the same construction as the first embodiment shown in Figure 1. The same components are denoted by the same references as those of the first embodiment to facilitate better understanding of the additional embodiment. The storage unit 15 can

store various data such as waveform data and various programs including the system control program or basic program, the waveform synthesizing program and other application programs. Normally, the ROM 11 provisionally stores these programs. However, if not, any program may be loaded into a hard disk or else in the storage unit 15. The loaded program is transferred to the RAM 13 to enable the CPU 10 to operate the inventive system of the musical sound generating apparatus. By such a manner, new or version-up programs can be readily installed in the system. For this purpose, a machine-readable media such as a CD-ROM (Compact Disc Read Only Memory) 51 is utilized to install the program. The CD-ROM 51 is set into a CD-ROM drive 52 to read out and download the program from the CD-ROM 51 into the storage unit 15 through the bus 12. The machine-readable media may be composed of a magnetic disk or an optical disk other than the CD-ROM 51.

[0069] A communication interface 53 is connected to an external server computer 54 through a communication network 55 such as LAN (Local Area Network), public telephone network and INTERNET. If the storage unit 15 does not reserve needed data or program, the communication interface 53 is activated to receive the data or program from the server computer 54. The CPU 10 transmits a request to the server computer 54 thorough the interface 53 and the network 55. In response to the request, the server computer 54 transmits the requested data or program to the apparatus. The transmitted data or program is stored in the hard disk of the storage unit 15 to thereby complete the downloading.

[0070] The inventive musical sound generating apparatus can be implemented by a personal computer which is installed with the needed data and programs. In such a case, the data and programs are provided to the user by means of the machine-readable media such as the CD-ROM 51 or a floppy disk. The machine-readable media contains instructions for causing the personal computer to perform the inventive musical sound generating method as described in conjunction with the previous embodiments. Otherwise, the personal computer may receive the data and programs through the communication network 55.

[0071] In the embodiments described above, the optional devices including the co-processor 17, DSP 21, sound source 22 are referred to as exemples, but the optional device is not limited to these devices. The present invention can be utilized in the application systems such as personal computer, electronic musical instrument, game machine and so on in which the musical sound is generated.

[0072] As shown in the foregoing, according to the present invention, various effects can be derived. It is possible to generate various musical sounds and to reduce a processing load required for musical sound generation. The musical sound can be generated at the optimum sample frequency for the configuration of the apparatus. The structure for generating the wave data of the musical sound can be significantly simplified. The quality of the generated musical sound can be retained even in low performance hardware. The musical sound can be generated according to the performance information even if the volume of the performance information becomes large.

Claims

1. A music apparatus having a central processor for generating a musical sound according to performance information, comprising:

   means for receiving the performance information;
   a waveform generator composed of a software program executable by the central processor in a variable operation mode dependently on a computation capacity of the central processor to create a digital waveform;
   means for changing the variable operation mode of the waveform generator according to the computation capacity available for operation of the waveform generator;
   the central processor operating the waveform generator in the variable operation mode as changed to create the digital waveform according to the received performance information; and
   means for generating the musical sound based on the created digital waveform.

2. The music apparatus according to claim 1, wherein the waveform generator is operable in the variable operation mode having a variable operation speed to create a digital waveform by successively computing sample values of the digital waveform, and is provisionally operated to carry out trial creation of a model digital waveform while measuring the computation capacity of the central processor in terms of the operation speed at which the trial creation is carried out, wherein the means for changing comprises means for determining the variable operation mode in terms of a sampling frequency comparable to the measured operation speed, and wherein the central processor actually operates the waveform generator to enable the same to successively compute sample values of the digital waveform at the determined sampling frequency.

3. The music apparatus according to claim 1, including means for detecting the computation capacity available for operation of the waveform generator by detecting whether or not an additional processor is available to assist the central processor in computation for executing the software program.

4. The music apparatus according to claim 3, wherein

the additional processor comprises a co-processor of the central processor.

5. The music apparatus according to claim 1, including means for detecting the computation capacity available for operation of the waveform generator by provisionally measuring the computation capacity of the central processor before the central processor executes the software program to operate the waveform generator.

6. The music apparatus according to claim 1, wherein the means for changing comprises means for changing the variable operation mode such that a first algorithm defining a method of creating the digital waveform is changed to a second algorithm simpler than the first algorithm when the computation capacity of the central processor decreases.

7. The music apparatus according to claim 1, wherein the means for changing comprises means for changing the variable operation mode according to the computation capacity of the central processor in terms of a variable sampling frequency by which the waveform generator variably creates samples of the digital waveform.

8. The music apparatus according to claim 1, wherein the means for changing comprises means for changing the variable operation mode such that a set of computation steps performed by the central processor to create the digital waveform is changed according to the computation capacity of the central processor.

9. A method of generating a musical sound by a central processor according to performance information, comprising the steps of:

receiving the performance information;
preparing a waveform generator composed of a software program executable by the central processor in a variable operation mode dependently on a computation capacity of the central processor to create a digital waveform;
changing the variable operation mode of the waveform generator according to the computation capacity available for operation of the waveform generator;
operating the waveform generator in the variable operation mode as changed to create the digital waveform according to the received performance information; and
generating the musical sound based on the created digital waveform.

10. A machine readable medium for use in a music apparatus having a central processing unit for generating a musical sound according to performance information, the medium containing instructions executable by the central processing unit for causing the music apparatus to perform a method comprising the steps of:

receiving the performance information;
preparing a waveform generator composed of a software program executable by the central processing unit in a variable operation mode dependently on a computation capacity of the central processing unit to create a digital waveform;
changing the variable operation mode of the waveform generator according to the computation capacity available for operation of the waveform generator;
operating the waveform generator in the variable operation mode as changed to create the digital waveform according to the received performance information; and
generating the musical sound based on the created digital waveform.

FIGURE 1

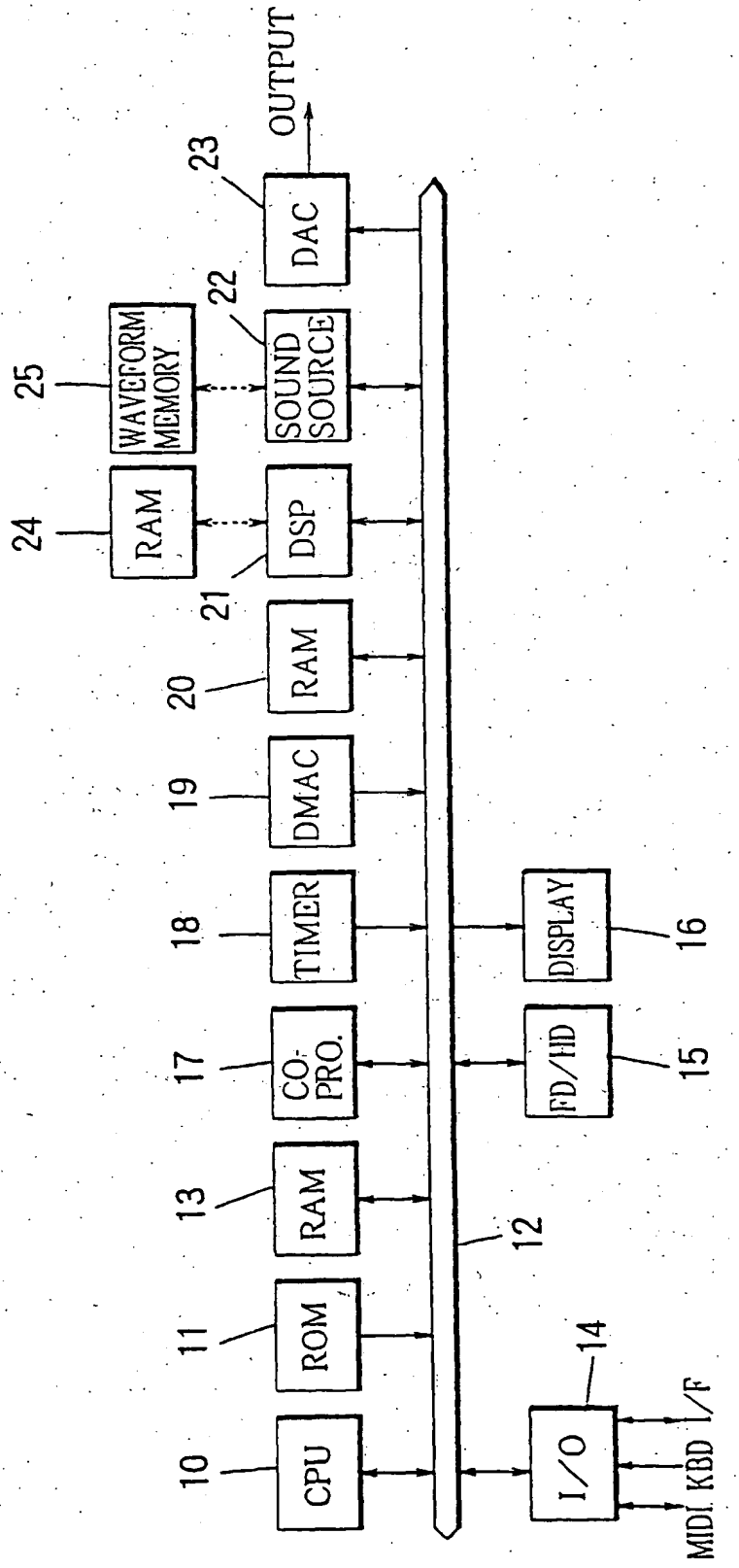


FIGURE 2

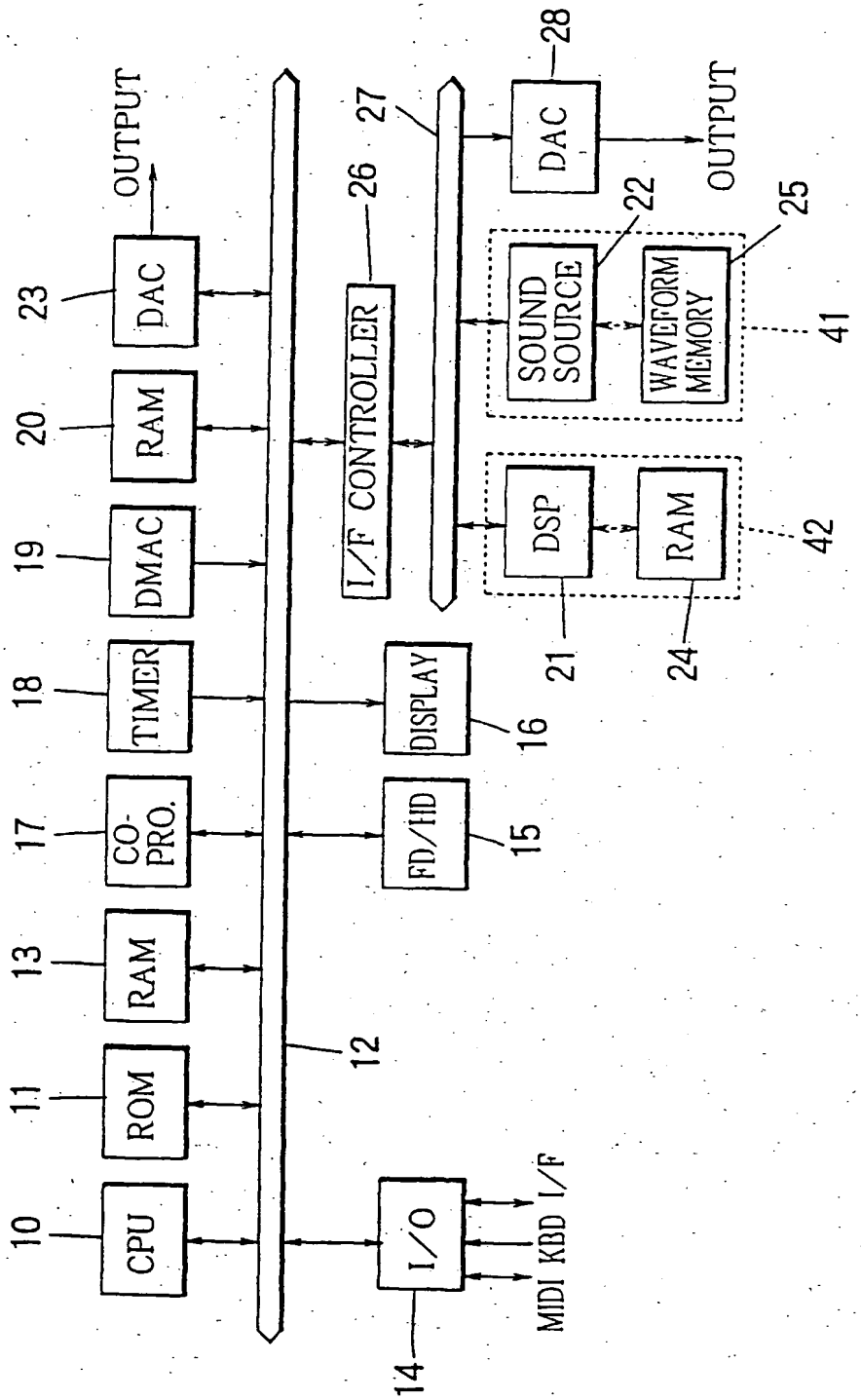


FIGURE 3

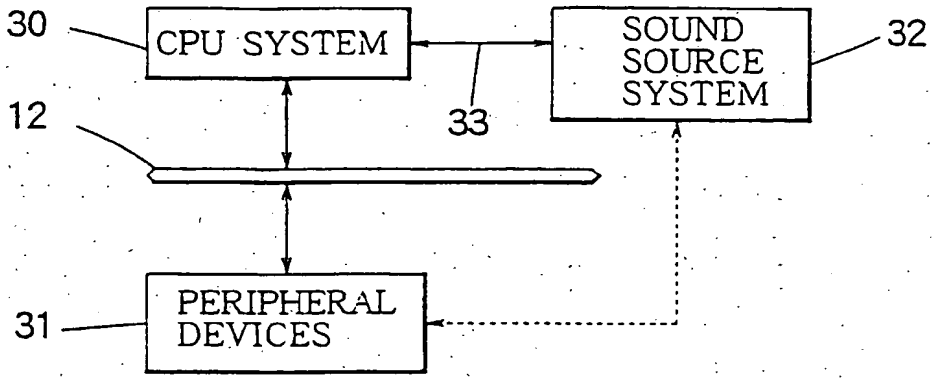


FIGURE 4A

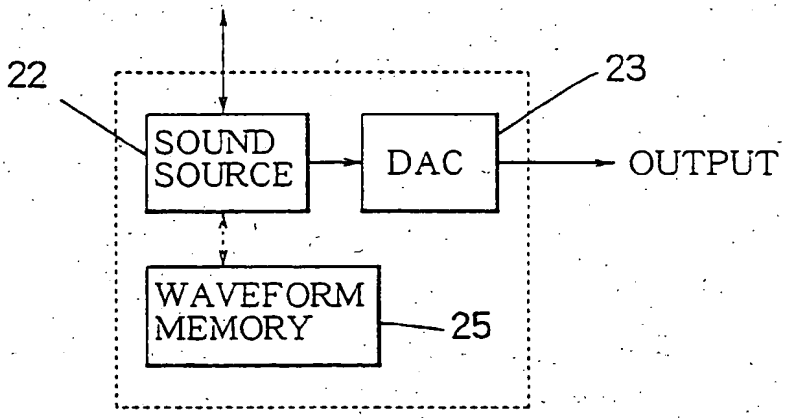


FIGURE 4B

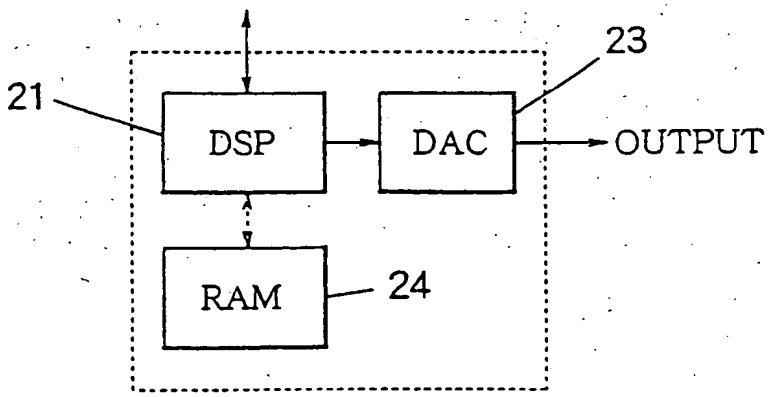


FIGURE 5

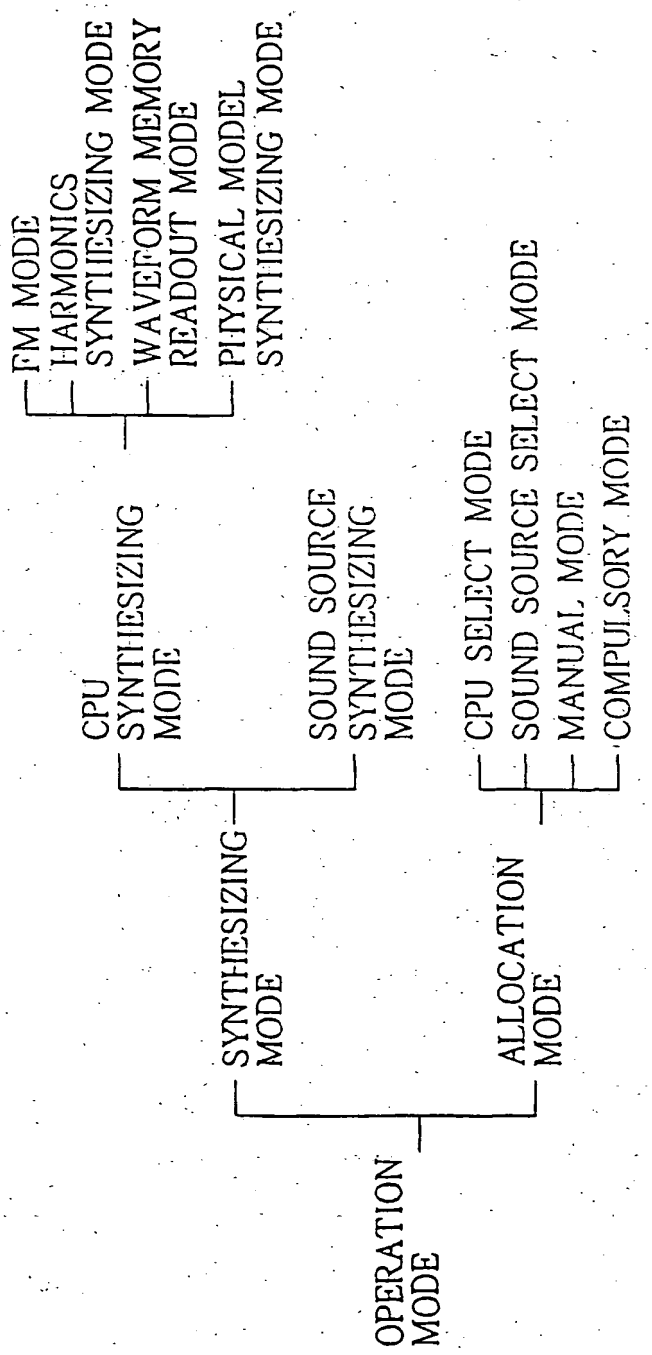


FIGURE 6

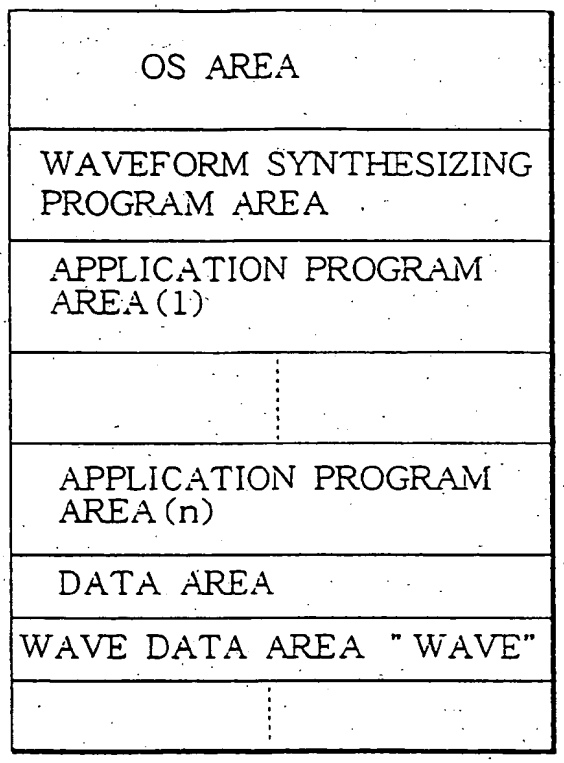


FIGURE 7

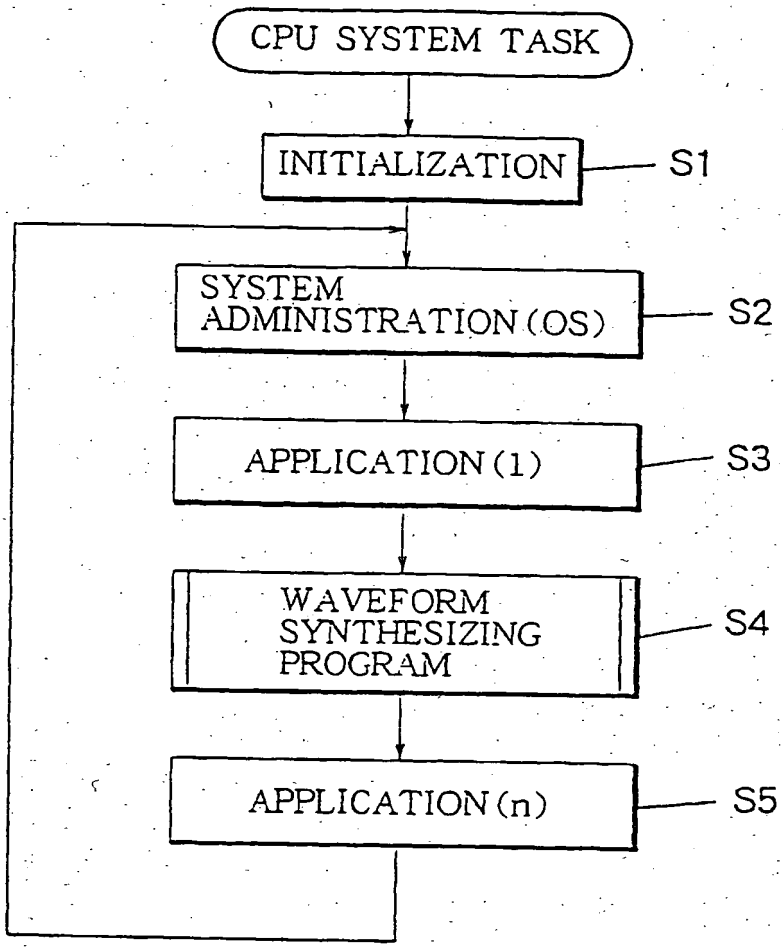


FIGURE 8

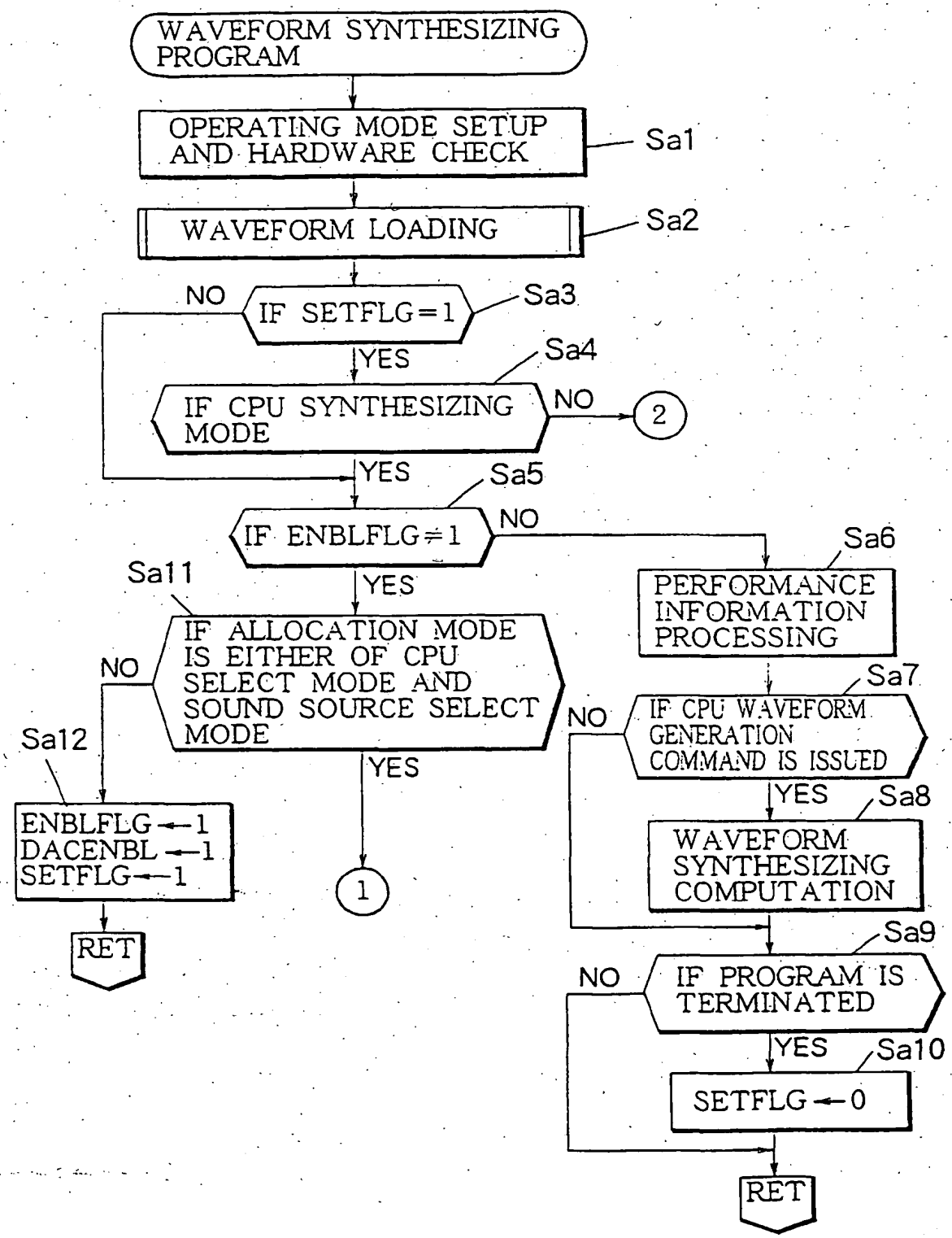


FIGURE 9

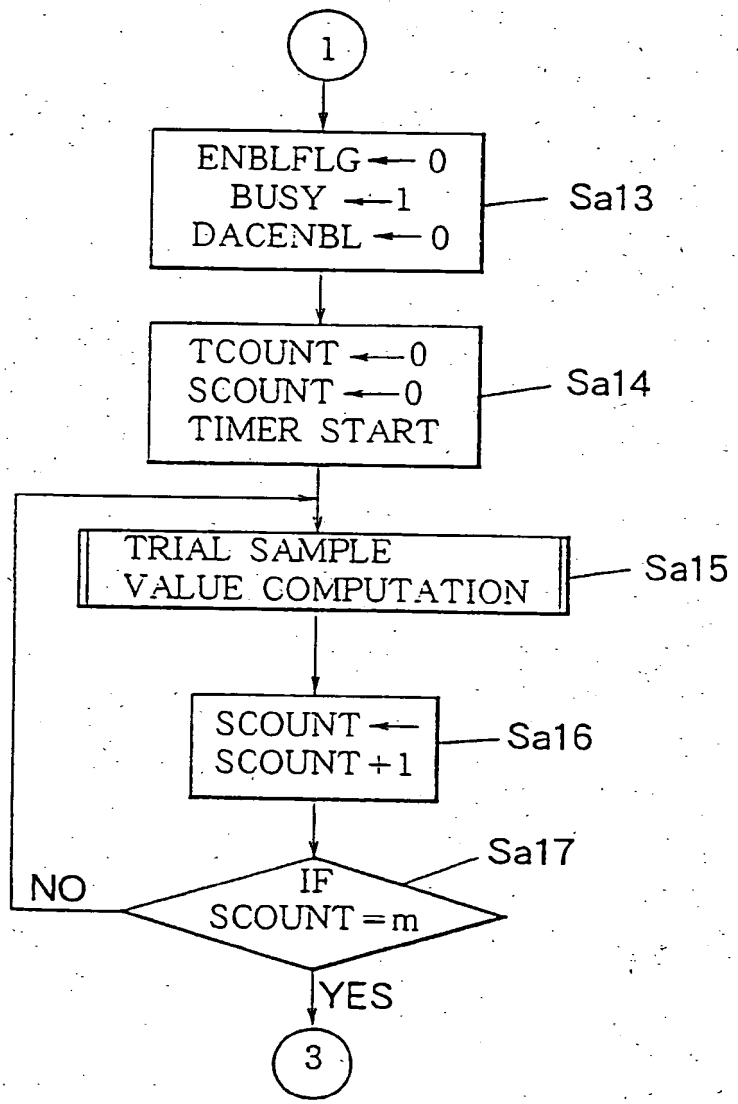


FIGURE 10

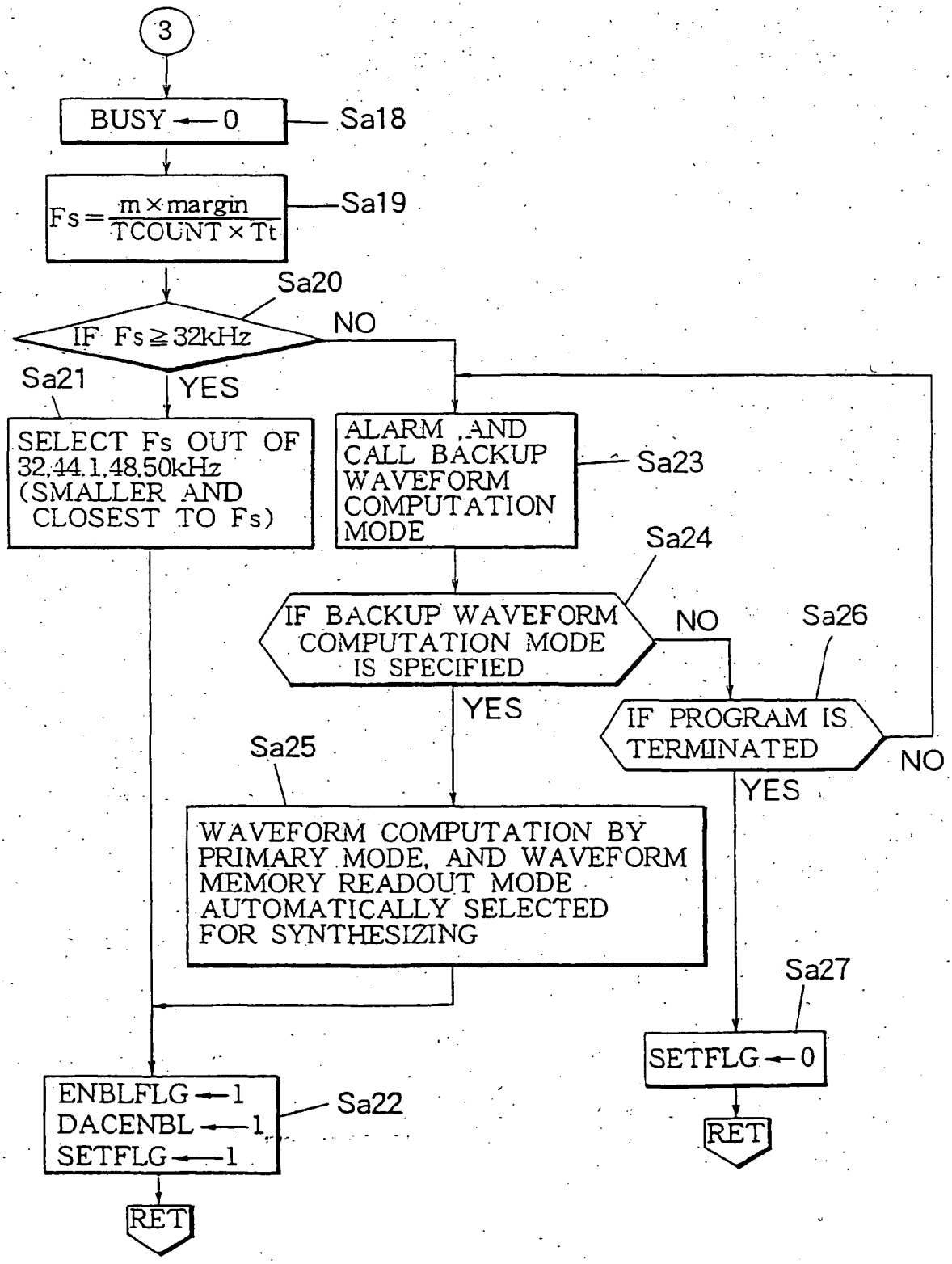


FIGURE 11

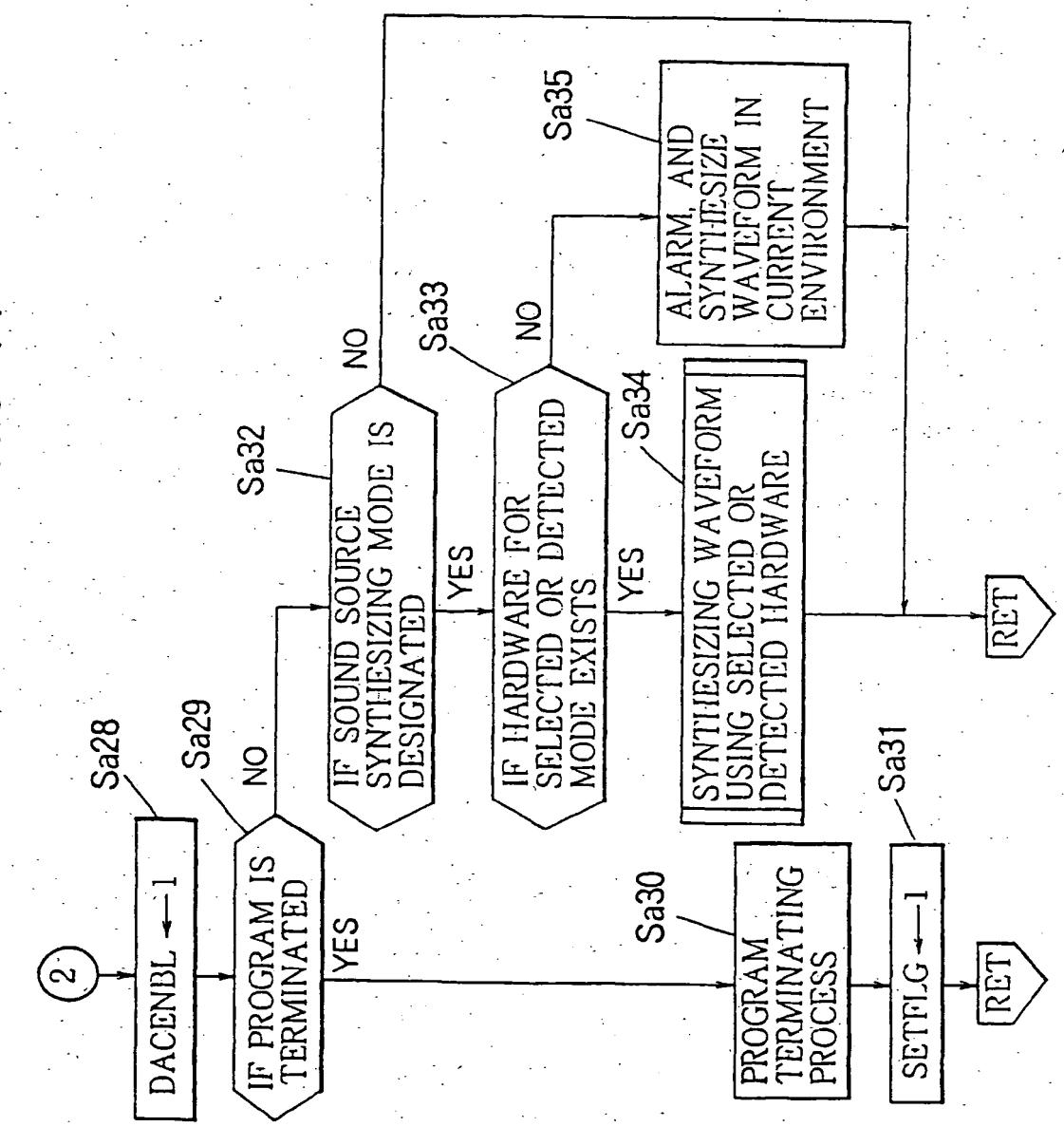


FIGURE 12

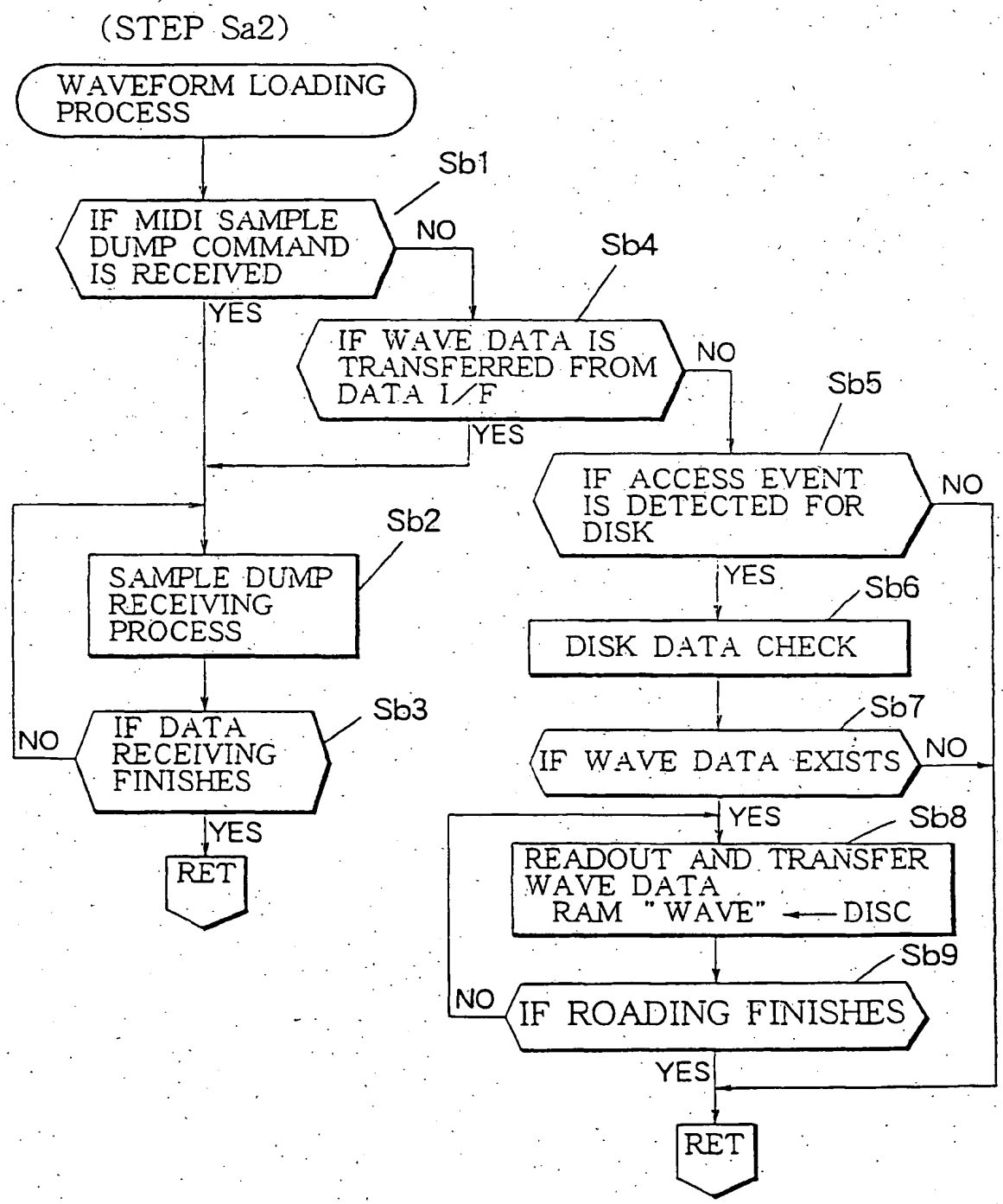


FIGURE 13

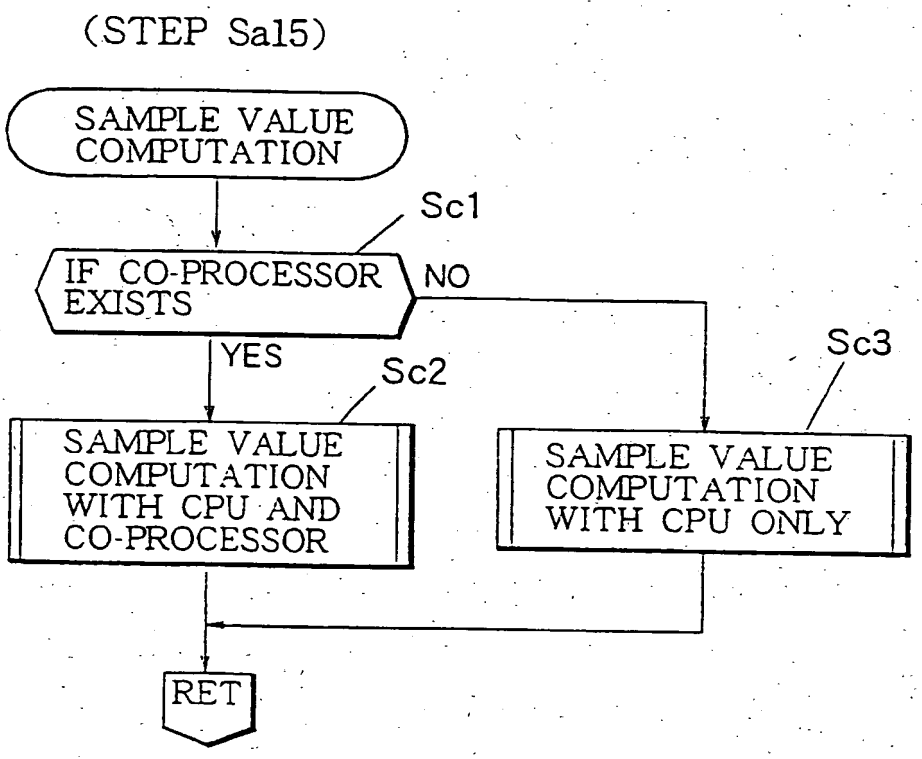


FIGURE 14

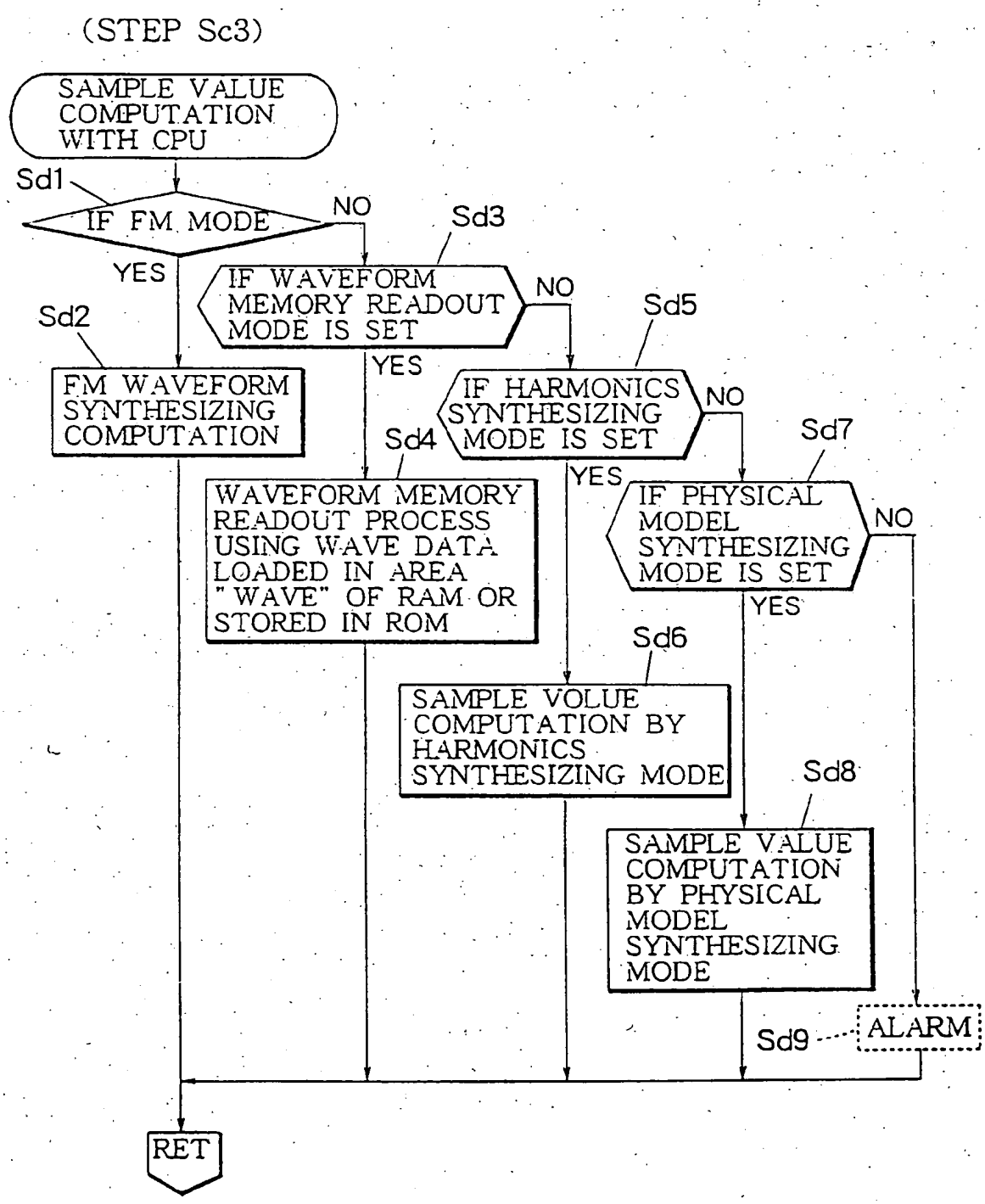


FIGURE 15

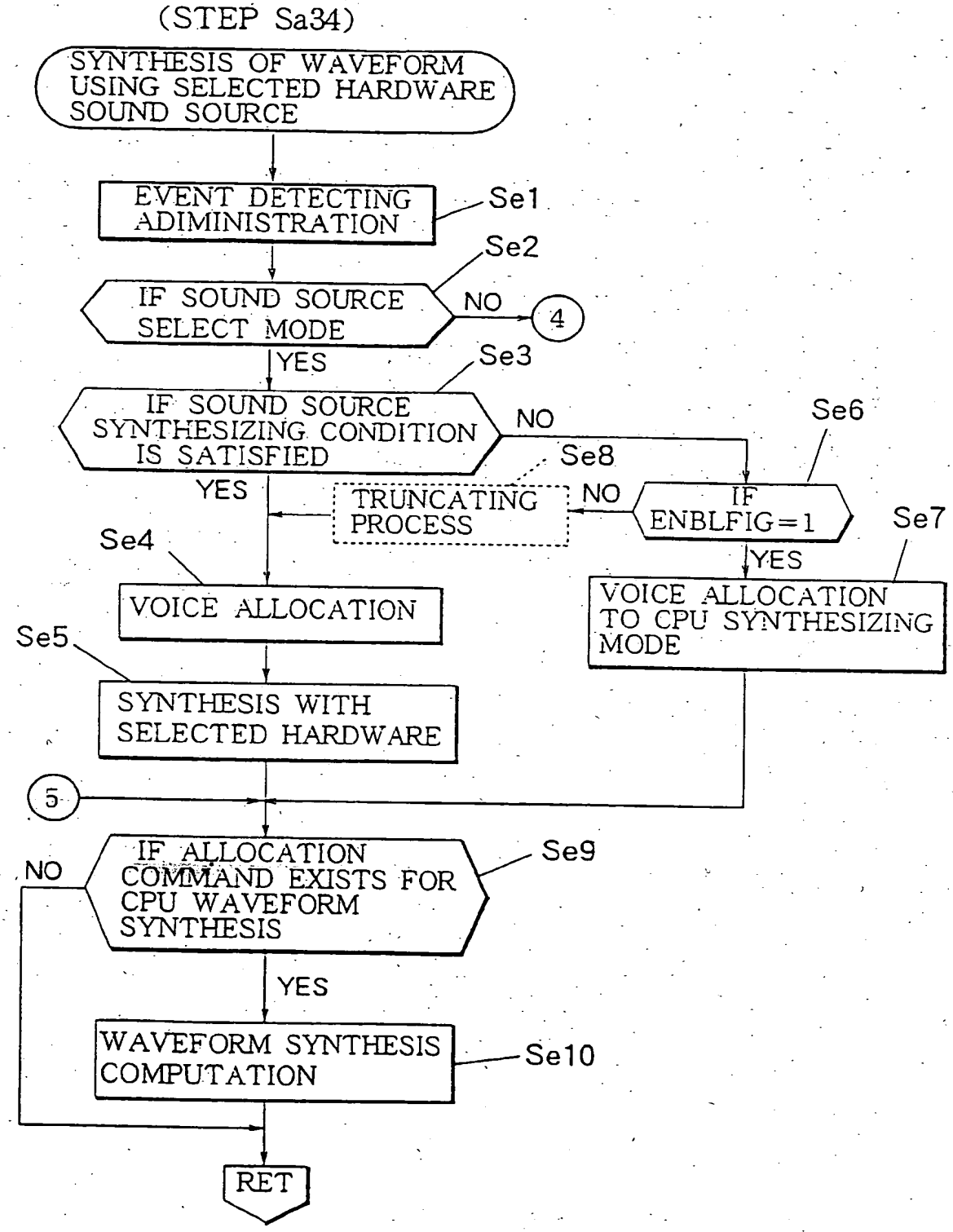


FIGURE 16

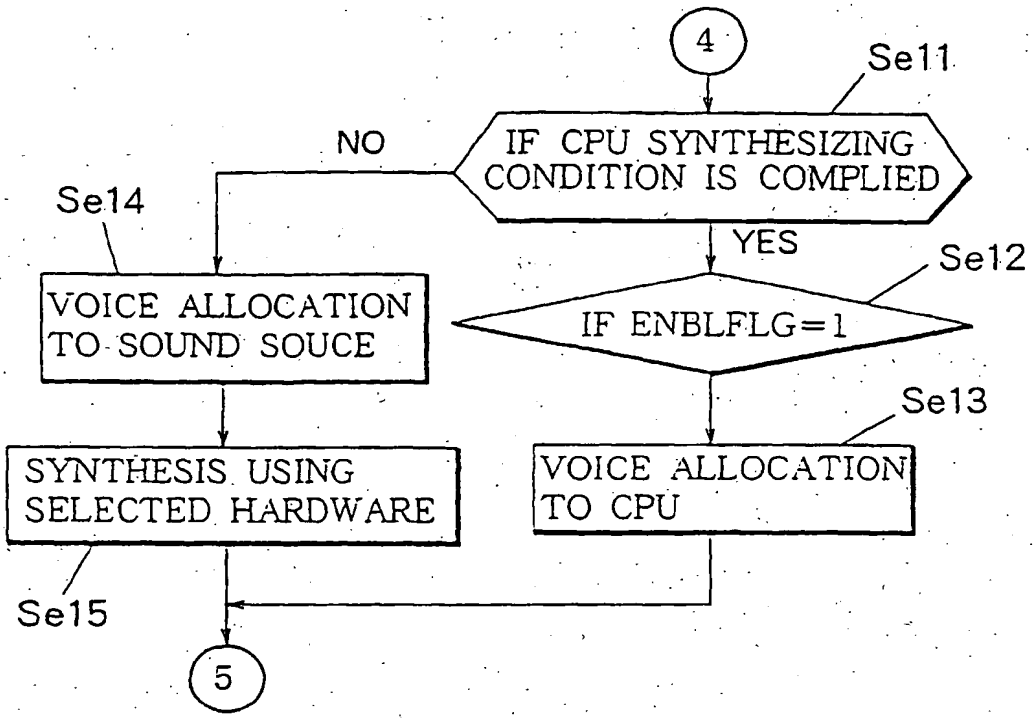


FIGURE 17

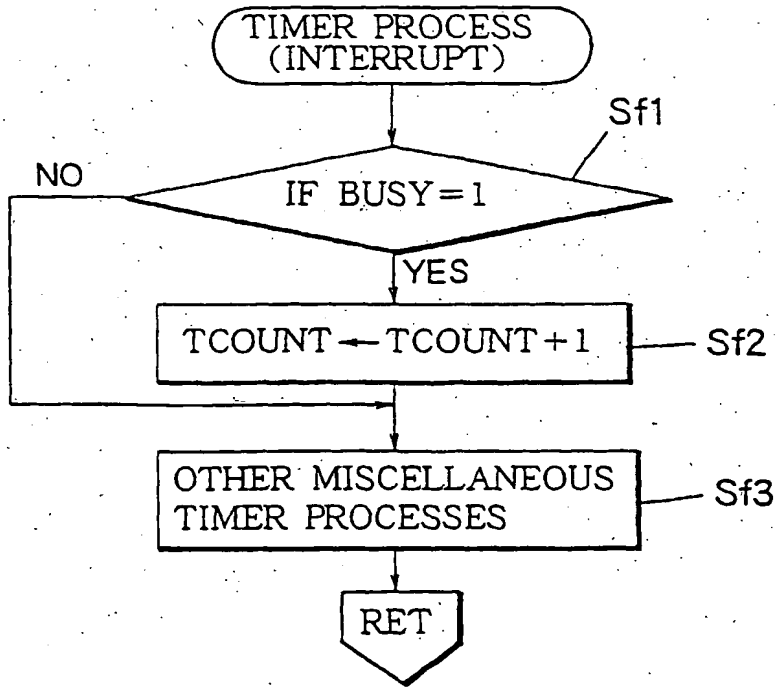


FIGURE 18

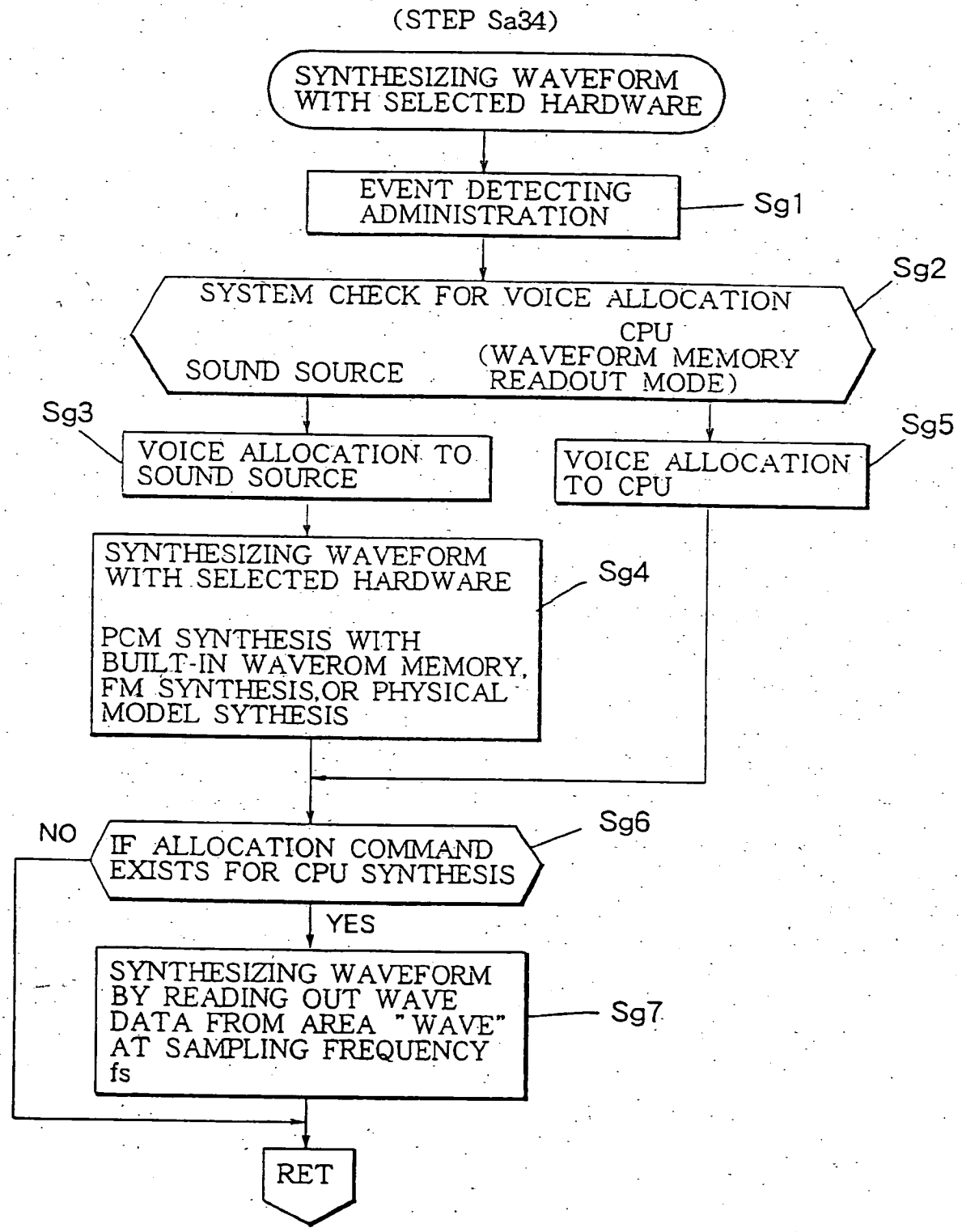


FIGURE 19

(STEP Sa34)

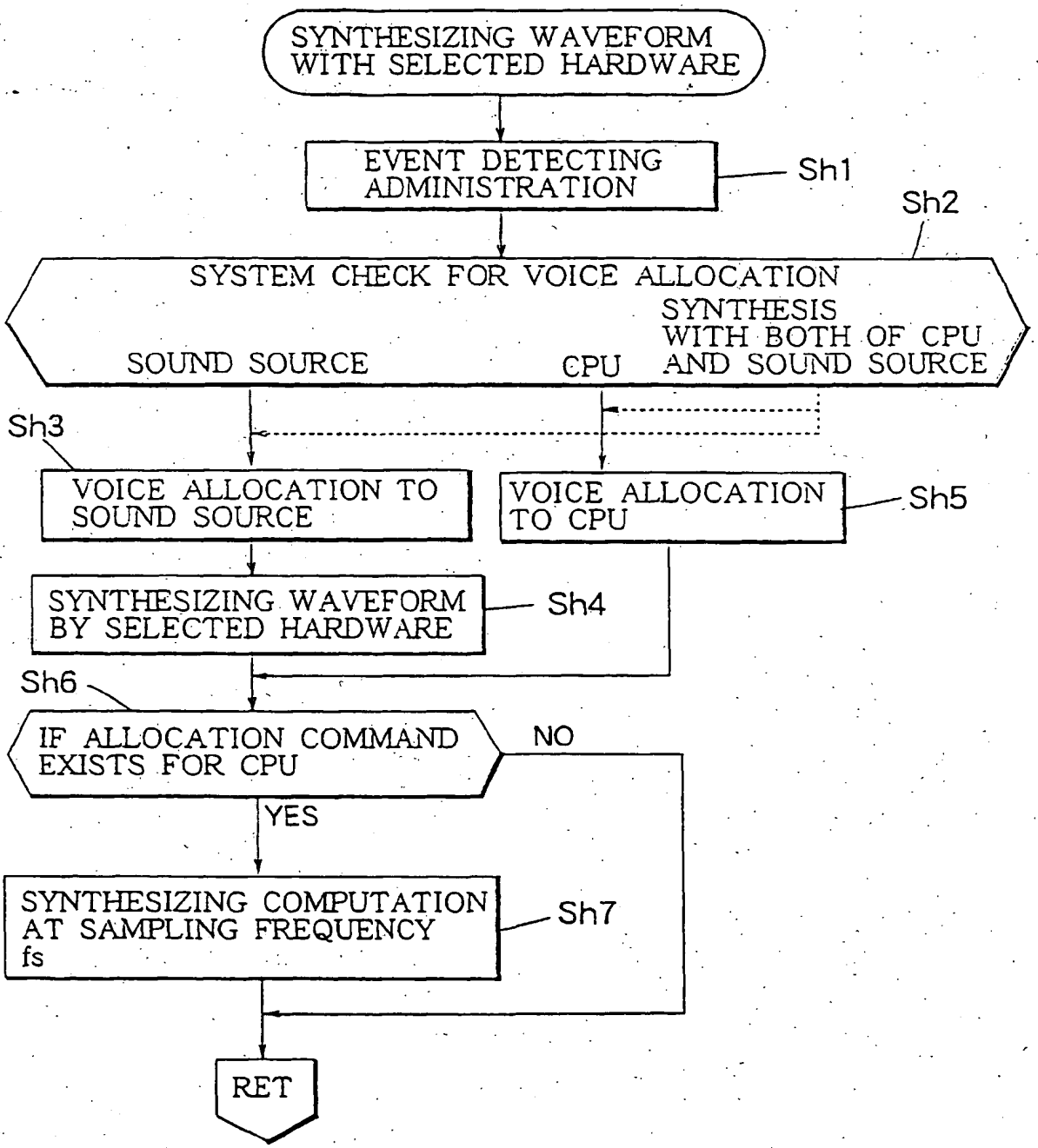


FIGURE 20

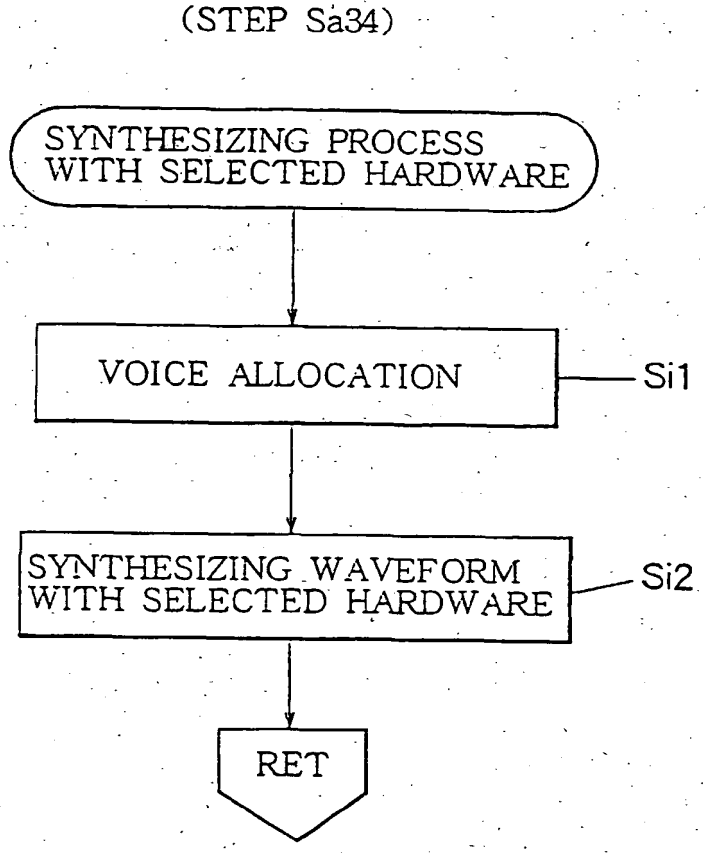


FIGURE 21

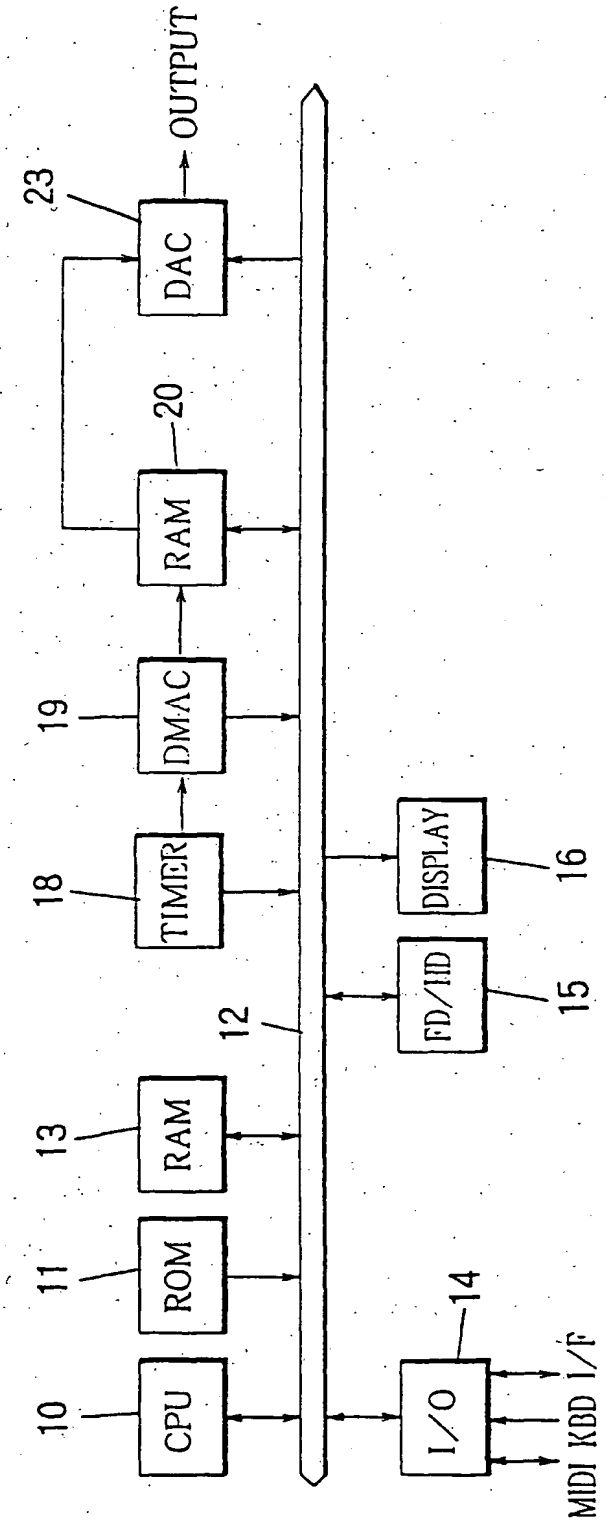


FIGURE 22

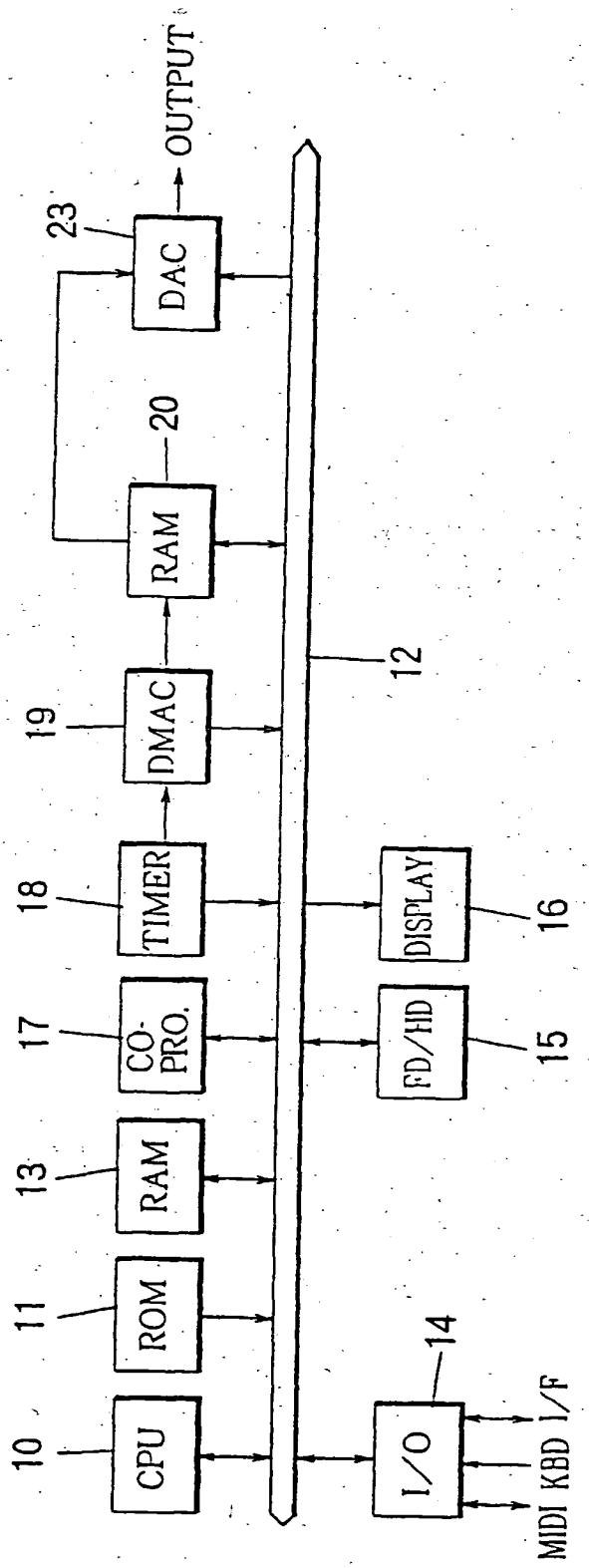


FIGURE. 23

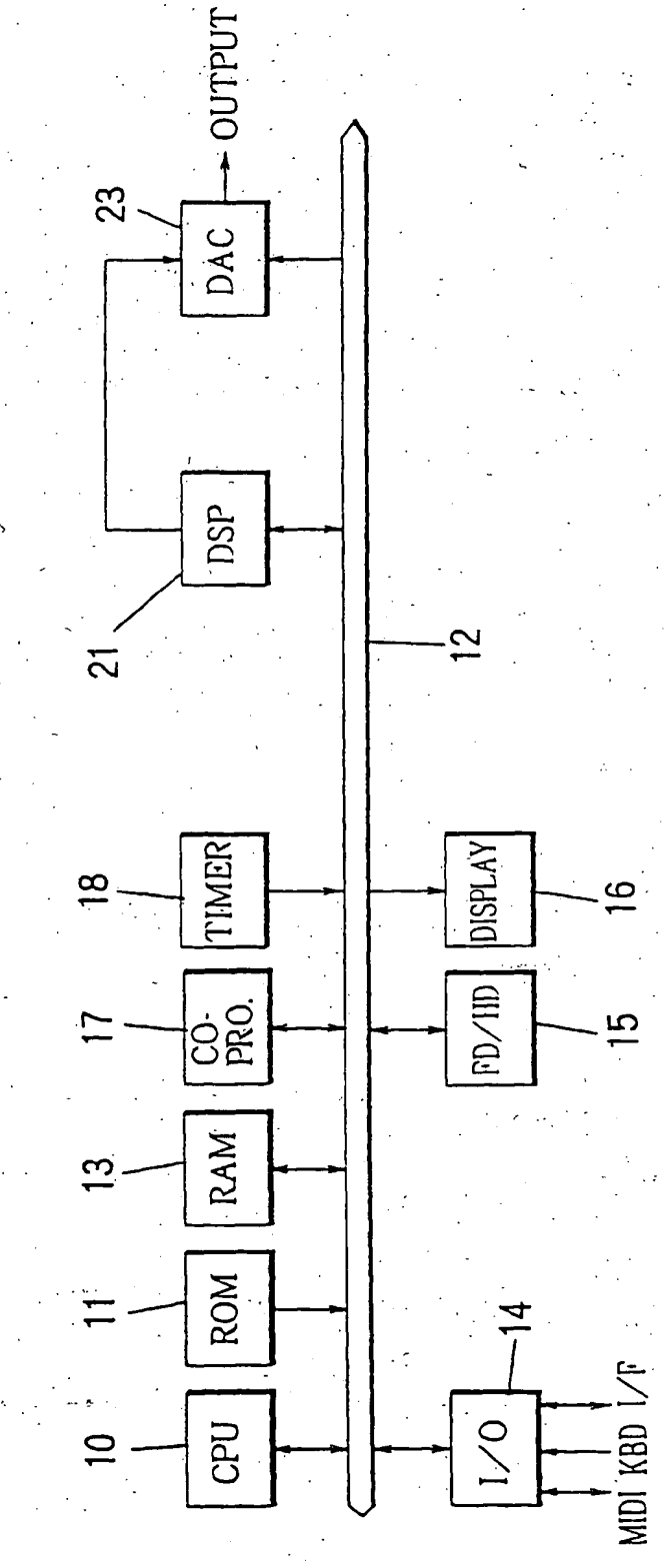


FIGURE 24

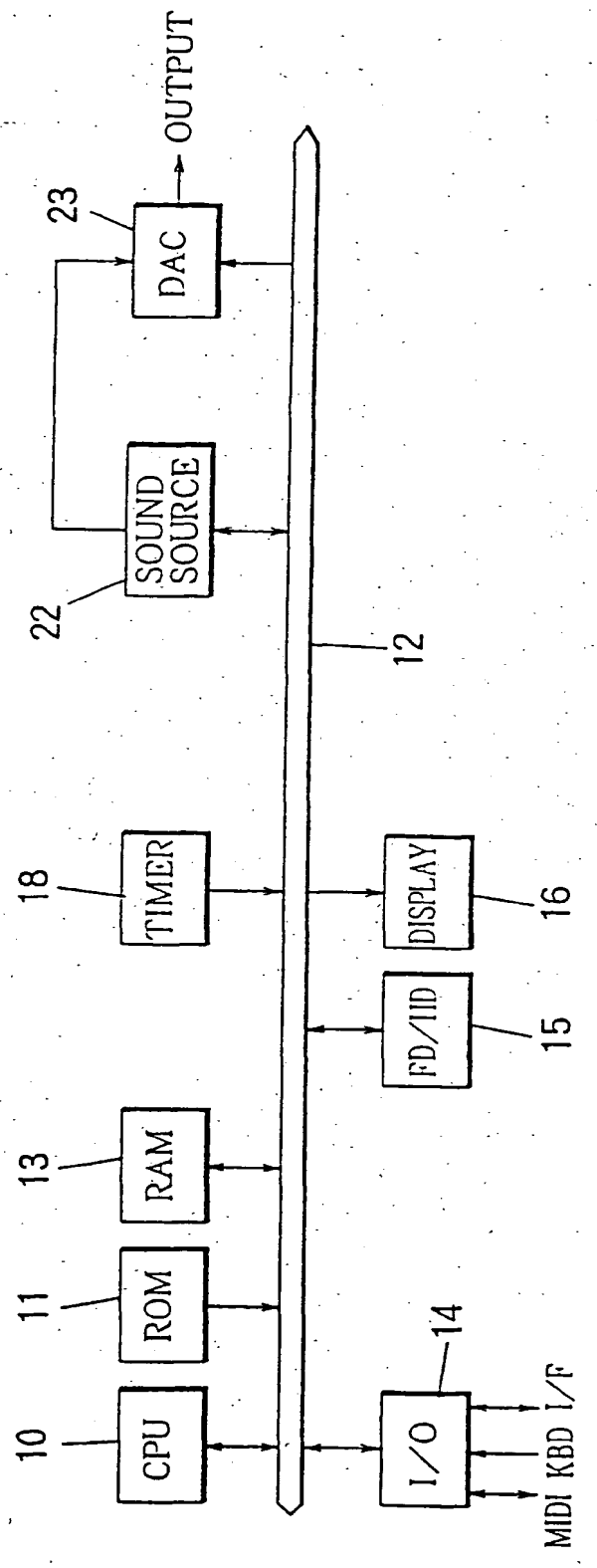
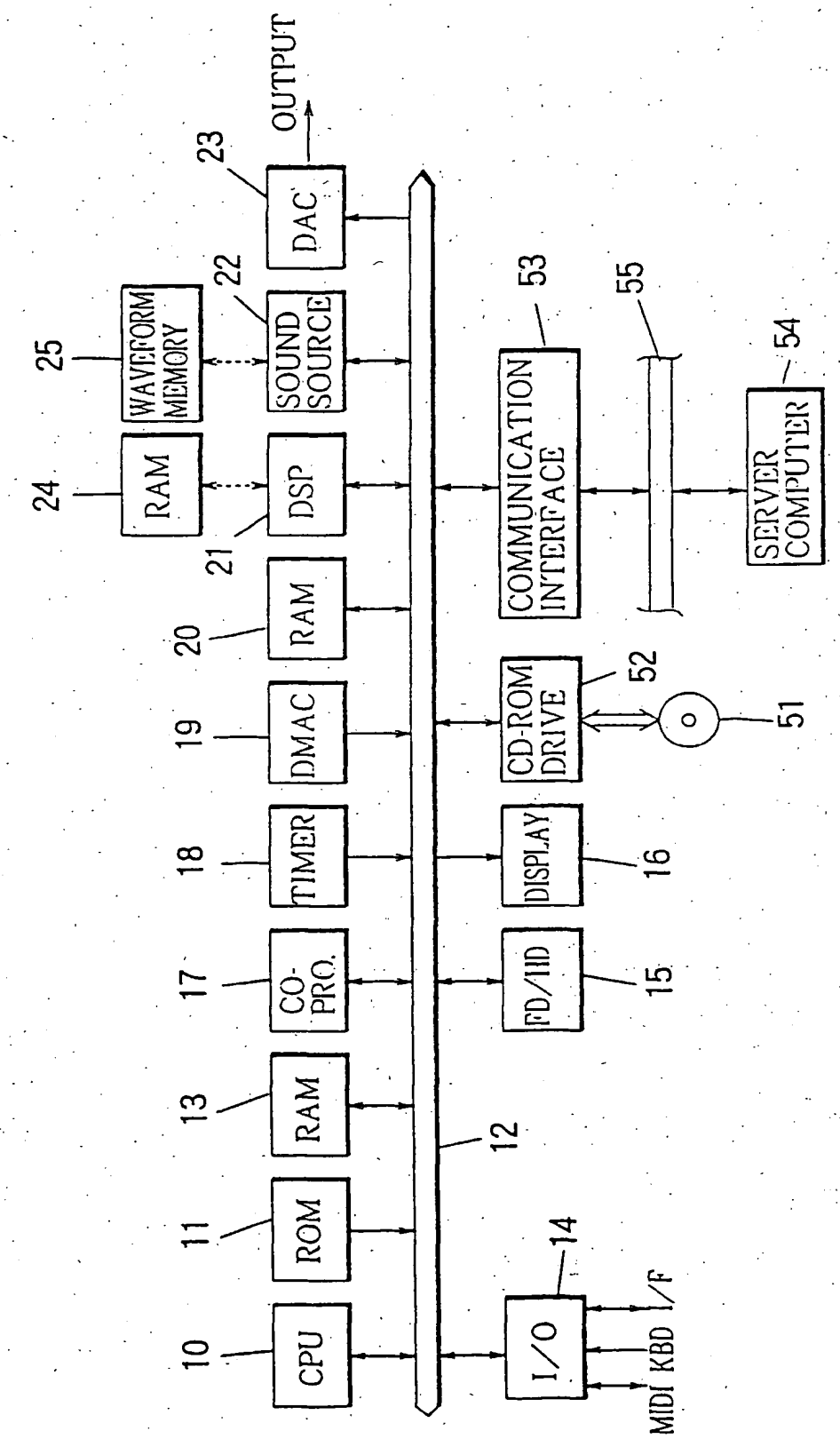


FIGURE 25



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 99 11 2006

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	WO 80 01215 A (FOX H ;SUTCLIFFE P (GB); MICROSKILL LTD (GB)) 12 June 1980 (1980-06-12) * page 12, line 3 - page 13, line 5 *	1,6,9,10	G10H7/00
A	US 5 376 752 A (LIMBERIS ALEXANDER J ET AL) 27 December 1994 (1994-12-27) * column 42, line 1 - line 25 * * column 43, line 8 - column 44, line 46; figure 1 *	1,3-6, 8-10	
A	US 5 119 710 A (TSURUMI KANEHISA ET AL) 9 June 1992 (1992-06-09) * column 3, line 5 - line 68 * * column 9, line 52 - column 11, line 54; figures 1,2 *	1,3,9,10	
A	"TWO METHODS OF SYNTHESISING MUSICAL SOUNDS BY MEANS OF MULTIPLE MICROPROCESSORS" RESEARCH DISCLOSURE, no. 201, 1 January 1981 (1981-01-01), page 52 XP002030867 ISSN: 0374-4353 * page 52, right-hand column, line 25 - line 29 *	1,2	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G10H

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	13 August 1999	Pulluard, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 99 11 2006

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

13-08-1999

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 8001215 A	12-06-1980	GB 2013386 A	08-08-1979
		EP 0013490 A	23-07-1980
		GB 2040537 A,B	28-08-1980
		JP 55500959 T	13-11-1980
		US 4438502 A	20-03-1984
US 5376752 A	27-12-1994	JP 2838645 B	16-12-1998
		JP 6308966 A	04-11-1994
		US 5657476 A	12-08-1997
US 5119710 A	09-06-1992	JP 5084919 B	03-12-1993
		JP 62208096 A	12-09-1987
		JP 1920314 C	07-04-1995
		JP 6042146 B	01-06-1994
		JP 62208098 A	12-09-1987
		JP 1912894 C	09-03-1995
		JP 6038192 B	18-05-1994
		JP 62208099 A	12-09-1987

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 087 372 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.03.2001 Bulletin 2001/13

(51) Int. Cl.[7]: **G10H 7/00**, G10H 1/18

(21) Application number: **00123550.6**

(22) Date of filing: **28.08.1997**

(84) Designated Contracting States:
DE GB IT

(30) Priority: **30.08.1996 JP 24694296**
30.08.1996 JP 24859296
14.01.1997 JP 1733397

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
97114933.1 / 0 827 132

(71) Applicant: **YAMAHA CORPORATION**
Hamamatsu-shi, Shizuoka-ken 430 (JP)

(72) Inventors:
- **Masuda, Hideyuki**
  Hamamatsu-shi, Shizuoka-ken, 430 (JP)
- **Isozaki, Yoshimasa**
  Hamamatsu-shi, Shizuoka-ken, 430 (JP)
- **Suzuki, Hideo**
  Hamamatsu-shi, Shizuoka-ken, 430 (JP)
- **Hirano, Masashi**
  Hamamatsu-shi, Shizuoka-ken, 430 (JP)

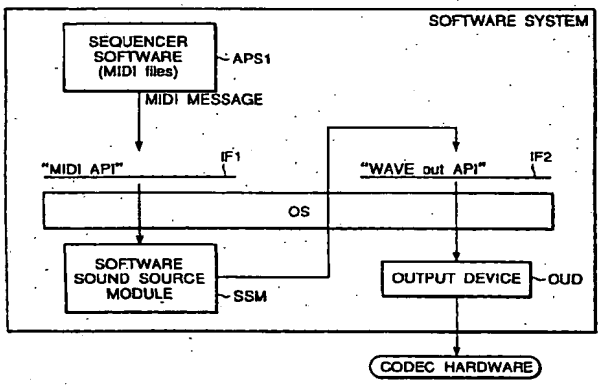
(74) Representative:
Kehl, Günther, Dipl.-Phys.
Patentanwaltskanzlei
Günther Kehl
Friedrich-Herschel-Strasse 9
81679 München (DE)

Remarks:
This application was filed on 27 - 10 - 2000 as a divisional application to the application mentioned under INID code 62.

(54) Sound source system based on computer software and method of generating acoustic data

(57) A sound source apparatus has software module used to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information (MIDI). The apparatus comprises a processor device (1) that periodically executes the software module to successively compute samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone. A detector device detects a load of computation imposed on the processor device (1) during the course of generating the musical tone. A controller device is operative according to the detected load and changes the variable sampling frequency to adjust a rate of computation of the samples.

FIG.1



EP 1 087 372 A2

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to a sound source system that combines music tone waveform generating modules made of software, and that generates music tone waveform data based on music tone waveform generating computation performed by each music tone waveform generating module. In addition, the present invention relates to a sound source waveform generating method that uses a general-purpose computation processing machine for executing a waveform computation algorithm so as to generate tone waveform data.

[0002] Conventionally, in order to generate a music tone according to a variety of music tone generating methods such as a waveform memory tone generating method and an FM tone generating method, a circuit for implementing the music tone generating method is constituted by dedicated hardware such as an LSI specifically designed for a sound source and a digital signal processor (DSP) that operates under the control of a fixed microprogram. The music tone generator constituted by the dedicated hardware is generically referred to as a hardware sound source hereafter. However, the hardware sound source requires dedicated hardware components, hence reduction of the product cost is difficult. It is also difficult for the hardware sound source to flexibly modify its specifications once the design has been completed.

[0003] For example, US 5,376,752 discloses a music synthesizer composed of a host processor module containing a CPU and a musical signal processor module (MSP), for performing a dynamic voice allocation. The host processor module provides a plurality of voice programs stored in the host memory. In the music signal processor module, the MSP memory stores a group of voice programs for active execution by the module. Namely, in the dynamic voice allocation, CPU never directly executes the program, but rather allocate the program to MSP, which in turn executes the program. To dynamically allocate a voice, a voice program must be moved from host memory in the host processor module to the MSP memory in the MSP module in real time, and without significant glitch in the audio output. The main CPU system consists of a microprocessor 50 such as a Motorola 68340. On the other hand, the MSP is a digital signal processor, which is provided separately from main CPU. The purpose for such a system is to enable dynamic voice allocation in a DSP based electronic music synthesizer between voices requiring differing DSP algorithms.

[0004] WO 80/01215 discloses an output processing system for a digital electronic musical instrument, which has a sound. source of four channels built by a DSP. The input control processor selects the microprogram and DSP executes the selected microprogram.

[0005] As another example for a hardware sound source, WO 9618995 discloses a PC audio system with wavetable cache. Disclosed is a hardware circuit, currently available as so called "sound card" for a personal computer (PC). Wavetable data is stored in a host computer's system memory and transferred in portions, as needed by the DSP, to a smaller, low cost cache memory incuded in the personal computer's audio circuit. The thus equipped PC-system has a lower overall cost.

[0006] US 5,410,099 discloses a channel assigning system for use in an electronic musical instrument, in which a predetermined number of simultaneously sounded musical tones of each part of a music peace can be secured and musical tones in a number greater than the predetermined number can be sounded. This is done by assigning channels other thanthe channels to be secured to newly performed musical tones.

[0007] A further hardware sound source is disclosed in 5,432,293. Described is a waveform generation device capable of reading waveform memory in plural modes. The device is directed to combining the processing of waveform data requiring high accuracy by employing an interpolation algorithm with the processing of waveform data requiring no such operation. This is done by using two processing modes, one involving an interpolation step, the other involving no such step. The selection of the modes is carried out in correspondence with the quality of the waveform data to be processed.

[0008] Finally, EP 0 536 644 discloses an electronic musical instrument having an algorithm selectable function. In the known instrument, some of the sound production channels for general purpose are predetermined for an exclusive use (only for a specific tone) by transferring desirable ones out of a plurality of sound production programs to each corresponding memory area of some (or at least one) of the sound production channels prior to synthesizing a tone or starting a performance. Some groups which are respectively comprised by the sound production channels which are defined exclusively for a desirable tone color, are also defined beforehand. After that, tones are generated according to a predetermined order of priority within a range of each group, based on supplied performance data.

[0009] Recently, as the computational performance of CPU has been enhancing, tone generators have been developed in which a general-purpose computer or a CPU installed on a dedicated tone generator executes software programs written with predetermined tone generation processing procedures to generate music tone waveform data. The tone generator based on the software programs is generically referred to as a software sound source hereafter.

[0010] Use of the hardware sound source in a computer system or a computer-based system presents problems of increasing the cost and decreasing the flexibility of modification. Meanwhile, the conventional soft-

ware sound sources simply replace the capabilities of the dedicated hardware devices such as the conventional tone generating LSI. The software sound source is more flexible in modification of the specifications after completion of design than the hardware sound source. However, the conventional software sound source cannot satisfy a variety of practical demands occurring during vocalization or during operation of the sound source. These demands come from CPU performance, system environment, user preferences and user settings. To be more specific, the conventional software sound sources cannot satisfy the demands for changing fidelity of an outputted music tone waveform (not only the change to higher fidelity but also to lower fidelity) and demands for changing the degree of timbre variation (for example, change from normal timbre variation to subtle timbre variation or vice versa).

[0011] Recently, an attempt has been made to generate tone waveform data by operating a general-purpose processor such as a personal computer to run software programs and to convert the generated digital tone waveform data through a CODEC (coder-decoder) into an analog music tone signal for vocalisation. The sound source that generates the tone waveform data by such a manner is referred to as the software sound source as mentioned before. Otherwise, the tone waveform data may be generated by an LSI dedicated to tone generation or by a device dedicated to tone generation having a digital signal processor (DSP) executing a microprogram. The sound source based on this scheme is referred to as the hardware sound source as mentioned before.

[0012] Generally, a personal computer runs a plurality of application software programs in parallel. Sometimes, a karaoke application program or a game application program is executed concurrently with a software sound source application program. This situation, however, increases a work load imposed on the CPU (Central Processing Unit) in the personal computer. Such an over load delays the generation of tone waveform data by the software sound source, thereby interrupting the vocalization of a music tone in the worst case. When the CPU is operating in the multitask mode, the above-mentioned concurrent processing may cause the tasks other than the tone generation task into a wait state.

[0013] In the hardware sound source, a waveform computation algorithm is executed by the DSP or the like to generate tone waveform data. The performance of the DSP for executing the computation has been enhanced every year, but the conventional tone waveform data generating method cannot make the most of the enhanced performance of the DSP.

SUMMARY OF THE INVENTION

[0014] It is therefore an object of the present invention to provide a sound source system based on computer software capable of reducing cost by generating a music tone by a software program without adding special dedicated hardware and, at the same time, capable of changing the load of a computation unit for computing music tone waveform and improving the quality of an output music tone.

[0015] It is another object of the present invention to provide a tone waveform data generating method that is capable of generating tone waveform data without interrupting the vocalization of a music tone even if the CPU load is raised high, and capable of, when the CPU is operating in the multitask mode, processing tasks not associated with the tone waveform generation without placing these tasks in a wait state.

[0016] It is still another object of the present invention to provide a tone waveform data generating method that makes a hardware sound source fully put forth its computational capability to provide the waveform output having higher precision than before.

[0017] To accomplish the above mentioned objects there is provided a sound source apparatus having a software module used to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information. In the inventive apparatus, a processor periodically executes the software module for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone. A detector device detects a load of computation imposed on the processor device during the course of generating the musical tone. A controller device operates according to the detected load for changing the variable sampling frequency to adjust a rate of computation of the samples.

[0018] Preferably, the controller device provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number.

[0019] Preferably, the processor device includes a delay device having a memory for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay device generates a write pointer for successively writing the samples into addresses of the memory and a read pointer for successively reading the samples from adresses of the memory to thereby create the delay corresponding to an address gap between the write pointer and the read pointer. The delay device is responsive to the fast sampling frequency to increment both of the write pointer and the read pointer by one address for one sample. Otherwise, the delay device is responsive to the slow sampling frequency to increment the write pointer by one address n times for one sample and to increment the read pointer by n addresses for one sample.

[0020] Preferably, the processor device includes a

delay device having a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay device successively writes the samples of the waveform of one mucical tone into addresses of one of the memory regions, and successively reads the samples from addresses of the same memory region to thereby create the delay. The delay device is operative when said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from addresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

[0021] Preferably, the processor device executes the software module composed of a plurality sub-modules for successively computing the waveform. The processor device is operative when one of the sub-modules declines to become inactive without substantially affecting other sub-modules during computation of the waveform for skipping execution of said one sub-module.

[0022] According to another embodiment, the sound source apparatus further comprises a provider device variably provides a trigger signal at a relatively slow rate to define a frame period between successive trigger signals, and periodically provides a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period. The processor device is resettable in response to each trigger signal and is operable based on each sampling signal to periodically execute the software module for successively computing a number of samples of the waveform within one frame. The controller device is operative according to the detected load for varying the frame period to adjust the number of the samples computed within one frame period. Further, a converter device is responsive to each sampling signal for converting each of the samples into a corresponding analog signal to thereby generate the musical tones.

[0023] According to another aspect of the invention, there is provided a method using a hardware processor and a software module to compute samples of a waveform in response to a sampling frequency for generating waveforms by means of a central processing unit for generating a musical tone according to performance information, as defined in claim 12. Favourable embodiments thereof are defined in the corresponding dependent claims 13 - 15.

[0024] According to yet another aspect of the invention, there is provided a machine readable media for use in a processor machine including a CPU, the media containing program instructions for causing the processor machine to carry out the above method. The media is defined in claim 16. Favourable embodiments thereof are defined in the corresponding dependent claims 17 - 19.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The above and other objects, features and advantages of the present invention will become more apparent from the accompanying drawings, in which like reference numerals are used to identify the same or similar parts in several views.

FIG. 1 is a schematic block diagram illustrating a software constitution of a sound source system practiced as a first preferred embodiment of the present invention;

FIG. 2 is a block diagram illustrating a general hardware constitution of the sound source system practiced as the first preferred embodiment of the present invention;

FIG. 3 is a diagram for explaining music tone generation processing performed by the sound source system of FIG. 1;

FIGS. 4A through 4C are a diagram for explaining overview of the music tone generation processing based on an FM sound source;

FIG. 5 is a diagram illustrating examples of basic waveform data selected from a basic waveform table;

FIG. 6 is a diagram illustrating a timbre register used for expanding timbre parameters of a music tone to be sounded through an assigned channel;

FIGS. 7A through 7C are a diagram illustrating a data format of music tone parameter VOICEj;

FIG. 8 is a diagram illustrating a MIDI-CH voice table for storing a voice number of music tone parameter VOICEn selectively set in each MIDI channel;

FIG. 9 is a flowchart indicating procedure of an initialization program executed by the CPU of the sound source system of FIG. 1;

FIG. 10 is a flowchart indicating procedure of a main program executed by the CPU after the initialization program of FIG. 9;

FIG. 11 is a flowchart indicating detailed procedure of a MIDI processing subroutine contained in the main routine of FIG. 10;

FIG. 12 is a flowchart indicating a continued part from the MIDI processing subroutine of FIG. 11;

FIG. 13 is a diagram illustrating an example of a format of a CH sequence register;

FIG. 14 is a flowchart indicating detailed procedure of a waveform computation processing subroutine contained in the main routine of FIG. 10;

FIG. 15 is a flowchart indicating a continued part from the waveform computation processing subroutine of FIG. 14;

FIG. 16 is a flowchart indicating detailed procedure of an FM computation processing subroutine for one channel;

FIG. 17 is a flowchart indicating detailed procedure of an operator computation processing subroutine for one operator;

FIG. 18 is a flowchart indicating a continued part from the operator computation processing subroutine;

FIG. 19 is a diagram illustrating a basic flow of an operator computation performed in the operator computation processing of FIGS. 17 and 18;

FIG. 20 is a flowchart indicating detailed procedure of a timbre setting processing subroutine contained in the main routine of FIG. 10;

FIG. 21 is a diagram illustrating a constitution of a software sound source system practiced as a second preferred embodiment of the present invention;

FIG. 22 is a diagram illustrating an operation timing chart of the software sound source system shown in FIG. 21;

FIG. 23 is a block diagram illustrating a processing apparatus having a tone waveform data generator implemented according to the tone waveform data generating method of the present invention;

FIG. 24 is a block diagram illustrating a constitutional example of a tube/string model section of a sound source model implemented according to the tone waveform data generating method of the present invention;

FIG. 25 is a block diagram illustrating a constitutional example of a timbre effect attaching section provided in the sound source model implemented according to the tone waveform data generating method of the present invention;

FIG. 26 is a block diagram illustrating a constitutional example of an exciter section provided in the sound source model implemented according to the tone waveform data generating method of the present invention;

FIG. 27 is a diagram illustrating a variety of data expanded in a RAM shown in FIG. 23;

FIG. 28 is a diagram illustrating details of control parameter VATONPAR necessary for computational generation of musical tones in the present invention;

FIG. 29 is a flowchart of an initialization program used in the present invention;

FIG. 30 is a flowchart of a main program in the present invention;

FIG. 31 is a flowchart of MIDI processing in the main program;

FIGS. 32A through 32C are a flowchart of physical model sound source key-on processing in the MIDI processing, a flowchart of other MIDI event processing and a flowchart of timbre setting processing activated by a user;

FIG. 33 is a flowchart of physical model parameter expansion processing in the timbre setting processing;

FIG. 34 is a part of a flowchart of waveform generation processing of a physical model sound source of the present invention;

FIG. 35 is the remaining part of the flowchart of the waveform generation processing of the physical model sound source of the present invention;

FIG. 36 is a flowchart of physical model computation processing in the tone waveform generation;

FIG. 37 is a flowchart of delay loop section computation processing in the physical model sound source computation processing;

FIG. 38 is a diagram for explaining a method of controlling a delay time length of a delay circuit of the physical model sound source;

FIG. 39 is a diagram for explaining a method of controlling a delay time length in the physical model sound source;

FIGS. 40A and 40B are a diagram illustrating a storage state of the control parameter VATONEPAR of each timbre;

FIG. 41 is a diagram illustrating a hardware constitution of a delay circuit in the physical model sound source associated with the present invention;

FIGS. 42A and 42B are a diagram for explaining an operation mode of the delay circuit shown in FIG. 41;

FIG. 43 is a diagram for explaining allocation of a delay memory area in a delay circuit included in the physical model sound source associated with the present invention; and

FIG. 44 is a diagram for explaining allocation of a delay circuit in a physical model sound source having a plurality of sound channels.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0026] This invention will be described in further detail by way of example with reference to the accompanying drawings. FIG. 1 shows a software constitution of a sound source system practiced as a first preferred embodiment of the present invention. As shown in the figure, this software sound source system is constituted to generate music tone waveform data based on an operating system (OS). It should be noted that FIG. 1 also shows CODEC hardware including a DAC (Digital to Analog Converter) for converting a digital music signal in the form of the music tone waveform data generated under control of the OS into an analog music tone signal.

[0027] Now, referring to FIG. 1, an APS1 is application software such as a sequencer software operable on real-time basis for sequentially generating performance information containing MIDI messages. The sequencer software APS1 has a plurality of MIDI files composed of MIDI data such as various event data and timing data for timing occurrence of the event data. The MIDI file is prepared for generating pieces of music. When one or more MIDI files are selected by the user, the MIDI data

is read sequentially from the selected files. Based on the read MIDI data, MIDI messages are sequentially generated according to the event data at real-time. Then, the sequencer software APS1 outputs the generated MIDI messages to a first interface IF1 which is a MIDI Application Interface or MIDI API arranged on the OS for MIDI message input.

[0028] The OS is installed with a driver defining a software sound source module SSM. This module is a program for generating music tone waveform data based on the MIDI messages inputted via the first interface IF1. The OS also has a second interface IF2 denoted by WAVE out Application Interface or WAVE out API for receiving the music tone waveform data generated by the software sound source module SSM. Further, the OS is installed with an output device OUD which is a software driver for outputting the music tone waveform data inputted via the second interface IF2. To be more specific, this output device OUD reads, via a direct memory access (DMA) controller, the music waveform data generated by the software sound source module SSM and temporarily stored in a storage device such as a hard disk, and outputs the read music waveform data to a predetermined hardware device such as a CODEC.

[0029] The MIDI messages outputted by the sequencer software APS1 are supplied to an input interface of the software sound source module SSM via the first interface IF1 and the OS. The software sound source module SSM performs music tone waveform data generation processing. In the present embodiment, the music tone waveform data is generated by FM tone generating based on the received MIDI messages. The generated music tone waveform data is supplied to the output device OUD via the second interface IF2 and the OS. In the output device OUD, the supplied music tone waveform data is outputted to the above-mentioned CODEC to be converted into an analog music tone signal.

[0030] Thus, the present embodiment allows, at the OS level, ready combination of the software sound source module SSM for generation music tone waveform data and the sequencer software APS1 which is the application software for outputting MIDI messages. This makes it unnecessary to add any hardware components dedicated to music tone waveform data generation, resulting in reduced cost.

[0031] FIG. 2 shows an overall hardware constitution for implementing the sound source system of the present embodiment. This system is implemented by a general-purpose personal computer. For the main controller of this system, a CPU 1 is used. Under the control of the CPU 1, the music tone waveform data generation processing by a software sound source program and processing by other programs are executed in parallel under multi-tasks.

[0032] Referring to FIG. 2, the CPU 1 is connected, via a data/address bus 19, to a MIDI interface (MIDI I/F) 12 for inputting MIDI messages from an external device and for outputting MIDI messages to an external device, a timer 16 for counting a timer interrupt time and other various times, a ROM (Read Only Memory) 2 for storing various control programs and table data, a RAM (Random Access Memory) 3 for temporarily storing a selected MIDI file, various input information, and computational results, a mouse 17 used as a pointing device, an alphanumeric keyboard 10 through which character information is mainly inputted, a display 5 composed of a large-sized LCD or a CRT for displaying various information, a hard disk drive 6 for driving a hard disk storing application programs, various control programs to be executed by the CPU 1 and various data, a DMA (Direct Memory Access) controller 14a, and a communication interface (I/F) 11 for transferring data between a server computer 102 via a communication network 101.

[0033] The DMA controller 14a directly reads the music tone waveform data generated by the music tone generation processing from an output buffer of the RAM 3 in direct memory access manner dependently on a free space state of a data buffer incorporated in a DAC 14b. The DMA controller 14a transfers the read music tone data to the data buffer of the DAC 14b for sound reproducing process. The analog music tone signal converted by the DAC 14b is sent to a sound system 18, in which the analog music tone signal is converted into a sound.

[0034] The hard disk of the hard disk drive 6 stores the above-mentioned OS, utility programs, software for implementing a software sound source that is the above-mentioned software sound source module SSM, and other application programs including the above-mentioned sequencer software APS1.

[0035] The output device OUD mentioned in FIG. 1 is equivalent to a module that sends the music tone data supplied from the software sound source module SSM via the above-mentioned second interface IF2 of the OS level to the DAC 14b. As mentioned above, the DMA controller 14a sends the music tone data to the DAC 14b in the direct memory access manner. The output device OUD is executed as interrupt processing by the DMA controller 14a under the control of the CPU 1.

[0036] The communication I/F 11 is connected to the communication network 101 such as a LAN (Local Area Network), the Internet, or a public telephone line. The communication I/F 11 is further connected to the server computer 102 via the communication network 101. If none of the above-mentioned programs and parameters are stored on the hard disk of the hard disk drive 6, the communication I/F 11 is used to download the programs and parameters from the server computer 102. A client computer (namely, the sound source system of the present embodiment) sends a command to the server computer 102 via the communication I/F 11 and the communication network 101 for requesting downloading of the programs and parameters. Receiv-

ing this command, the server computer 102 distributes the requested programs and parameters to the client computer via the communication network 101. The client computer receives these programs and parameters via the communication I/F 11, and stores the received programs and parameters in the hard disk of the hard disk drive 6, upon which the downloading operation is completed. In addition, an interface for transferring data directly between an external computer may be provided.

[0037] The following is an overview of the music tone generation processing based on FM tone generating by the software sound source module SSM with reference to FIGS. 3 through 6. When the sequencer software APS1 is started, MIDI messages are supplied to the software sound source module SSM. To be more specific, the MIDI messages are supplied to a software sound source interface via the first interface IF1 and the OS. Accordingly, the software sound source module SSM generates a music tone parameter VOICEj based on voice data in the form of a voice number assigned to a MIDI channel of the supplied MIDI message. The voice number represents a particular timbre of the music tone. The MIDI channel may corresponds to a particular performance part of the music piece. The SSM loads the generated music tone parameter VOICEj into a timbre register corresponding to a sound channel which is designated or allocated for sounding of the particular performance part of the music piece.

[0038] FIG. 6 shows a timbre register group provided to the sound channels. If 32 number of the sound channels are allocated for example, this timbre register group has 32 number of timbre registers TONEPARk (k = 1 to 32). It will be apparent that the number of sound channels is not limited to 32 but may be set to any value according to the computational performance of the CPU 1.

[0039] Referring to FIG. 6, if the sound channel concerned is channel n, the music tone parameter VOICEj is stored in a area for storing the music tone parameter VOICEj in the timbre register TONEPARn. In other words, the timbre register group composed of these timbre registers TONEPARk provides a part of the software sound source interface of the software sound source module SSM.

[0040] It should be noted that, in addition to the music tone parameter VOICEk, these timber registers TONEPARk store data TM indicating a time at which the software sound source module SSM has received a MIDI message corresponding to the music tone parameter VOICEk. The data TM provides information for determining time positions of key-on and key-off operations within a predetermined frame of period.

[0041] Referring to FIG. 3, the software sound source module SSM is basically started by a trigger signal which is set for each frame having a predetermined time length, under the control of the CPU 1. The SSM executes the music tone generation processing based on the MIDI messages supplied within a frame immediately before the trigger, according to the music tone parameter VOICEn stored in the timbre register TONEPARn. For example, as shown in FIG. 3, the music tone generation processing based on the MIDI messages supplied within a preceding frame from time t1 to time t2 is executed in a succeeding frame from time t2 to time t3.

[0042] When the music tone waveform data for one frame has been generated by the music tone generation processing, the generated music tone waveform data is written to the output buffer of the RAM 3. Reproduction of the written data is reserved in the output device OUD. This reservation in the OUD is equivalent to the outputting of the generated music tone waveform data from the software sound source module SSM to the second interface IF2 (WAVE out API) of the OS level.

[0043] The output device OUD reads the music tone waveform data, a sample by sample, from the output buffer reserved for the reproduction in the immediately preceding frame, and outputs the data to the DAC 14b. For example, as shown in FIG. 3, the music tone waveform data generated in the frame from time t2 to time t3 and written to the output buffer for reserved reproduction is read in a next frame from time t3 to time t4 for the sound reproduction.

[0044] The following is an overview of the music tone generation processing based on music tone parameter VOICEn. In this embodiment, the music tone generation processing is based on FM tone generating as shown in FIGS. 4A through 4C. FIG. 4A through FIG. 4C show three different music tone generating methods. As shown in the figures, the music tone generation based on FM tone generating is performed by combining two types of operation blocks or operators, namely, an operator called a carrier and an operator called modulator. The different number of combined operators and the different connection sequences (connection modes) are used according to the type and quality of the music tone waveform to be generated. Systematic connection scheme of these operators is called an algorithm.

[0045] The operator herein denotes a block that provides a unit in which tone creation or music tone generation processing is performed. To be more specific, from various basic waveform data used for the tone creation, one piece of basic waveforms shown in FIG. 5 for example is selected according to a wave select parameter WSEL and is read based on input data such as pitch data and modulation data. If the input data includes two types of data such as the pitch data and the modulation data, the basic waveform data is read out based on a result obtained by adding these two pieces of data together. Then, the amplitude of this one piece of waveform data is adjusted, and the adjusted data is outputted. The operation block in which these operations are performed is called the operator. Among the operators, the carrier denotes an operator for generating a basic music tone waveform. The modulator denotes an operator for modulating the carrier, namely

for generating modulation data for modulating the carrier. It should be noted that the algorithm is not limited to the three types shown in FIGS. 4A through 4C.

[0046] The following explains a data format of the above-mentioned music tone parameter VOICEj. FIGS. 7A through 7C show the data format of the music tone parameter VOICEj. FIG. 7A shows the data format of the music tone parameter VOICEj, FIG. 7B shows a data format of each operator data OPmDATAj shown in FIG. 7A, and FIG. 7C shows a data format of each operator buffer OPBUFm shown in FIG. 7B.

[0047] As shown in FIG. 7A, the music tone parameter VOICEj is composed of key-on data KEYONj indicating key-on and key-off by "1" and "0" respectively, frequency number FNOj (actually represented by a phase rate) determined by pitch information included in a MIDI message of a corresponding note-on event, algorithm designation data ALGORj for designating one of the above-mentioned algorithms, volume data VOLj determined according to volume set to a MIDI channel concerned. The volume is set by control change #7 event of the MIDI message, for example. The music tone parameter further contains touch velocity data VELj determined according to touch velocity information in the MIDI message concerned, and operator data OPkDATAj (j = 1 to m) made up of a buffer for holding data necessary for computing music tone generation in each of the constituent operators and the results of this computation.

[0048] It should be noted that the music tone parameter VOICEj simultaneously has two types of data, one type read from the ROM 2, RAM 3, or the hard disk and the other type determined according to the data in the MIDI message. The data determined according to the MIDI message includes the key-on data KEYONj, the frequency number FNOj, the volume data VOLj, and the touch velocity data VELj. The data read from the ROM 2 and so on includes the algorithm designation data ALGORj and the operator data OPkDATAj.

[0049] As shown in FIG. 7B, each operator data OPkDATAj is composed of sampling frequency designation data FSAMPm for designating a sampling frequency used in operator m, frequency multiple data MULTm providing a parameter for substantially setting a frequency ratio between operators (actually, a parameter for designating an integer multiple for varying the above-mentioned frequency number FNOj), feedback level data FBLm indicating a feedback level (namely, a degree of feedback modulation), wave select data WSELm for selecting basic waveform data to be used by operator m from various pieces of basic waveform data (described with reference to FIG. 5) stored in the ROM 2, total level data TLm for setting the output level (varying with the above-mentioned touch velocity data VELj) of a music tone waveform to be generated in the operator m, envelope parameter EGPARm composed of various type of data (for example, attack time, decay time, sustain level, and release time) for determining the envelope of a music tone waveform to be generated in the operator m, data MSCm indicating other parameters (for example, velocity and depth of vibrato and tremolo, and various key scaling coefficients), operator priority data OPPRIOm indicating priority of operator m (for example, priorities of start and stop of the waveform generating computation in each operator), and buffer OPBUFm for storing the results of the music tone waveform generating computation in operator m.

[0050] The sampling frequency designation data FSAMPm contains integer value f higher than "0". This integer value f allows the sampling frequency FSMAX (for example, 44.1 kHz) in standard mode to be multiplied by 2^{-f}. For example, if f = 0, a music tone waveform in operator m is generated at the sampling frequency FSMAX of the standard mode; if f = 1, a music tone waveform in operator m is generated at the sampling frequency of FSMAX/2.

[0051] The operator priority data OPPRIOm contains data (for example, numbers indicating the order by which waveform computing operations are performed) indicating the priority of the waveform computation processing in all operators k (k = 1 to m). According to this priority data, the priority by which each operator is activated is determined for the waveform computation processing. Alternatively, the performance and load states of the CPU 1 are checked to determine the operators to be activated. If this check indicates that the CPU 1 has no more capacity for performing tone generation processing, the computation processing of the operators of lower priorities may be left out. In the present embodiment, the priorities of the computation processing are set according to timbre applied to the music tone. Alternatively, the priorities may be set according to MIDI channels for example. Namely, the priorities set by some reference may be selected for use at sounding. For example, if the priorities are not set according to the timbre, the operator priority data OPPRIOm may be determined based on the timbre parameter expanded in the above-mentioned timbre register TONEPARn. The operator priority data OPPRIOm may be handled also as to determine the setting that operator m is to be used or not.

[0052] In the present embodiment, the sampling frequency can be set for each operator m by the above-mentioned sampling frequency designation data FSAMPm. Alternatively, the sampling frequency may be set differently for the two types of the operators, the carrier and the modulator. For example, the carrier may be set to the above-mentioned frequency FSMAX and the modulator may be set to 1/2 of the FSMAX. In this case, the contents of the algorithm of the timbre parameter concerned are checked and the sampling frequency may be accordingly set for the operators with which the timbre parameter is combined. Alternatively, the load state of the CPU 1 is checked and the sampling frequency may be accordingly increased or decreased.

[0053] As shown in FIG. 7C, the buffer OPBUFm is

composed of operator-on parameter OPONm indicating by "1" that the waveform computation is performed by operator m (namely, operator m is on), phase value buffer PHBUFm for storing a phase value obtained by performing phase computation on the result of the waveform computation performed by operator m, feedback output value buffer FBm for storing a feedback output value obtained by the feedback sample computation of the above-mentioned waveform computation processing, modulation data input buffer MODINm for storing modulation data (this data is used in the above-mentioned phase computation processing), operator output value buffer OPOUTm for storing the music tone waveform (namely the output value) generated by operator m, and EG state buffer EGSTATEm for storing the EG parameters obtained by the computation processing (hereafter referred to as AEG computation processing) for computing amplitude controlling EG of the above-mentioned waveform computation processing.

[0054] FIG. 8 shows a MIDI-CH voice table for storing voice data representative of a timbre selectively set for each MIDI channel or for each performance part of the music piece. In the present embodiment, the voice data is denoted by a voice number of music tone parameter VOICEn.

[0055] As shown in FIG. 8, in the present embodiment, 16 MIDI channels are provided. Different timbres can be set to different MIDI channels corresponding to different performance parts. Consequently, the sound source system of the present embodiment can generate a maximum of 16 types of timbres. This MIDI-CH voice table lists the voice numbers of the timbres assigned to the sound channels, namely the voice numbers contained in the above-mentioned music tone parameters VOICEn.

[0056] The MIDI-CH voice table is allocated at a predetermined area in the RAM 3. The table data, namely the voice numbers, are stored beforehand on the hard disk or the like in correspondence with the selected MIDI file. The user-selected MIDI file is loaded into a performance data storage area allocated at a predetermined location in the RAM 3. At the same time, the table data corresponding to the loaded MIDI file is loaded into the MIDI-CH voice table. Alternatively, the user can arbitrarily set the MIDI-CH voice table from the beginning or can change the table after standard voice numbers have been set to the music piece. MIDI messages are sequentially generated by the sequencer program APS1 and the generated MIDI messages are recognized by the software sound source module SSM. The software sound source module SSM then searches the MIDI-CH voice table for the voice number assigned to the MIDI channel of the MIDI message concerned. For example, if the MIDI channel of the MIDI message concerned is "2HC," the voice number stored at the second location VOICENO2 in the MIDI-CH voice table is selected.

[0057] When voice number j is found, the software sound source module SSM generates music tone parameter VOICEj as described above. To be more specific, the software sound source module SSM reads the basic data from the ROM 2 and determines other parameters from the MIDI message concerned to generate the music tone parameter VOICEj shown in FIGS. 7A through 7C. Then, the software sound source module SSM expands the generated music tone parameter VOICEj in a timbre register TONEPARn corresponding to the sound channel among the plurality of timbre registers shown in FIG. 6.

[0058] As descrived above, the inventive sound source apparatus has the operation blocks OPs (shown in FIGS. 4A through 4C) composed of softwares used to compute waveforms for generating a plurality of musical tones through a plurality of sound channels according to performance information in the form of the MIDI messages. In the inventive apparatus, a setting device sets an algorithm (shown in FIGS. 4A through 4C) which determines a system of the software sound source module SSM composed of selective ones of the operation blocks OPs systematically combined with each other to compute a waveform specific to one of the musical tones. A designating device including the MIDI API shown in FIG. 1 responds to the performance information for designating one of the channels to be used for generating said one musical tone. A generating device including the CPU 1 allocates the selective operation blocks to said one channel and systematically executes the allocated selective operation blocks according to the algorithm so as to compute the waveform to thereby generate said one musical tone through said one channel.

[0059] Preferably, the setting device sets different algorithms which determine different systems corresponding to different timbres of the musical tones. Each of the different systems is composed of selective ones of the operation blocks which are selectively and sequentially combined with each other to compute a waveform which is specific to a corresponding one of the different timbres.

[0060] Preferably, the setting device comprises a determining device that determines a first system combining a great number of operation blocks and corresponding to a regular timbre and that determines a second system combining a small number of operation blocks and corresponding to a substitute timbre, and a changing device operative when a number of operation blocks executable in the channel is limited under said great number and over said small number due to a load of the computation of the waveform for changing the musical tone from the regular timbre to the substitute timbre so that the second system is adopted for the channel in place of the first system.

[0061] Preferably, the setting device comprises an adjusting device operative dependently on a condition during the course of generating the musical tone for adjusting a number of the operation blocks to be allo-

cated to the channel.

[0062] Preferably, the adjusting device comprises a modifying device that modifies the algorithm to eliminate a predetermined one of the operation blocks involved in the system so as to reduce a number of the operation blocks to be loaded into the channel for adjustment to the condition.

[0063] Preferably, the adjusting device operates when the condition indicates that an amplitude envelope of the waveform attenuates below a predetermined threshold level for compacting the system so as to reduce the number of the operation blocks.

[0064] Preferably, the adjusting device operates when the condition indicates that an output volume of the musical tone is tuned below a predetermined threshold level for compacting the system so as to reduce the number of the operation blocks.

[0065] Preferably, the adjusting device operates when the condition indicates that one of the operation blocks declines to become inactive in the system without substantially affecting other operation blocks of the system for eliminating said one operation block so as to reduce the number of the operation blocks to be allocated to the channel.

[0066] Preferably, the generating device comprises a computing device responsive to a variable sampling frequency for executing the operation blocks to successively compute samples of the waveform in synchronization to the variable sampling frequency so as to generate the musical tone, and a controlling device that sets the variable sampling frequency according to process of computation of the waveform by the operation blocks.

[0067] Preferably, the generating device comprises a computing device responsive to a variable sampling frequency for executing the operation blocks to successively compute samples of the waveform in synchronization to the variable sampling frequency so as to generate the musical tone, and a controlling device for adjusting the variable sampling frequency dependently on a load of computation of the waveform during the course of generating the musical tone.

[0068] Preferably, the generating device comprises a computing device responsive to a variable sampling frequency for executing the operation blocks to successively compute samples of the waveform in synchronization to the variable sampling frequency so as to generate the musical tone, and a controlling device for adjusting the variable sampling frequency according to result of computation of the samples during the course of generating the musical tone.

[0069] The following explains the control processing to be performed by the sound source system thus constituted, with reference to FIGS. 9 through 20. FIG. 9 is a flowchart showing the procedure of an initialization program to be executed by the CPU 1 in the sound source system of the present embodiment. The initialization program is executed when the user turns on the power to the sound source system, or presses a reset switch thereof. First, system initialization such as resetting ports and clearing the RAM 3 and a video RAM in the display 5 is performed (step S1). Next, the OS program is read from the hard disk of the hard disk drive 6 for example, and the OS program is loaded in a predetermined area in the RAM 3 so as to run the OS program (step S2). Then, the process goes to the execution of a main program.

[0070] FIG. 10 is a flowchart indicating the procedure of the main program to be executed by the CPU 1 after execution of the initialization program. This main program is the main routine of the software sound source module SSM. First, the area containing the timbre register group shown in FIG. 6 in the RAM 3 to be used by the software sound source module SSM is cleared. At the same time, the various types of basic data (for example, the various pieces of basic waveform data shown in FIG. 5) stored in the hard disk of the hard disk drive 6 are loaded in a predetermined area in the RAM 3 (step S11). Next, basic graphic operation is performed to display information according to the progression of processing and to display menu icons to be selected mainly by the mouse 7 (step S12).

[0071] Then, the sound source module SSM checks to see whether any of the following triggers has taken place (step S13).

Trigger 1: the sequencer software APS1 has been started for supplying a MIDI message to the software sound source module SSM.

Trigger 2: an internal interrupt signal (a start signal) for starting execution of the waveform computation processing by the SSM has been generated by a software timer.

Trigger 3: a request has been made by the CODEC hardware for transferring the music tone waveform data from the output buffer to a buffer in the CODEC hardware.

Trigger 4: the user has operated the mouse 7 or the keyboard 8 and the corresponding operation event has been detected.

Trigger 5: the user has terminated the main routine and the corresponding operation event has been detected.

[0072] In step S14, the CPU 1 determines which of the above-mentioned triggers 1 through 5 has taken place. If the trigger 1 has been taken place, the software sound source module SSM passes control by the CPU 1 to step S16, in which a MIDI processing subroutine is executed. If the trigger 2 has been taken place, the software sound source module SSM passes control to step S17, in which a waveform computation processing subroutine is executed. If the trigger 3 has taken place, the process goes to step S18, in which the music tone waveform data is transferred from the output buffer to the buffer of the CODEC hardware. If the trigger 4 has

taken place, the software sound source module SSM passes control to step S19, in which a timbre setting processing subroutine is executed especially if a timbre setting event has occurred; if another event has occurred, corresponding processing is performed in step S20. If the trigger 5 has taken place, the software sound source module SSM passes control to step S21, in which end processing such as returning the screen of the display 5 to the initial state provided before the main program was started. Then, any of the steps S16 through S21 has been ended, the software sound source module SSM passes control to step S12 to repeat the above-mentioned operations.

[0073] FIGS. 11 and 12 are flowcharts indicating the detailed procedure of the MIDI processing subroutine of step S16. First, the software sound source module SSM checks to see whether a MIDI event (a MIDI message) has been inputted via the software sound source interface API of the software sound source module SSM (step S31). When a MIDI message is outputted from the sequencer software APS1, the outputted MIDI message is converted in a predetermined manner by the first interface IF1 and the OS. The converted MIDI message is then transferred to a MIDI event buffer allocated at a predetermined area in the RAM 3 via the software sound source interface API. When this transfer is made, the software sound source module SSM determines that the trigger 1 has taken place, thereby passing control by the CPU 1 from step S15 to step S16. The processing operations so far are performed in the preparation processing of step S20 in the main routine of FIG. 10. In step S31, the software sound source module SSM monitors the event occurrence by checking the MIDI event buffer.

[0074] Next, in step S32, the software sound source module SSM determines whether the MIDI event is a note-on event. If the MIDI event is found a note-on event, the software sound source module SSM passes control to step S33; if not, the SSM passes control to step S40 shown in FIG. 12. In step S33, the SSM decodes the note-on event data and stores resultant note-number data, velocity value data and part number data (namely, the MIDI channel number) into registers NN, VEL, and p, respectively. Further, the SSM stores the data about the time at which the note-on event should take place into a register TM allocated at a predetermined position in the RAM 3. Hereafter, the contents of the registers NN, VEL, p, and TM are referred to as note number NN, velocity VEL, part p, and time TM, respectively.

[0075] In step S34, the software sound source module SSM determines whether velocity VEL is lower than a predetermined value VEL1 and whether volume data VOLp is lower than a predetermined value VOL1. The VOLp denotes the volume data of the part p stored in area VOLp allocated at a predetermined area in the RAM 3. This VOLp is changed by the control change #7 event of the MIDI message as explained with reference to FIG. 7A. The change is performed in the miscellaneous processing of step S20 when the control change #7 event has taken place. In step S34, if VEL ≤ VEL1 and VOL ≤ VOL1, the regular timbre allotted to the part p is replaced by a substitute timbre of an algorithm having a small number of operators, namely a small total number of carriers and modulators. That is, the voice number stored in VOICEp of the part p in the above-mentioned MIDI-CH voice table is replaced by the voice number of the music tone parameter VOICE having an alternate algorithm (step S35). If VEL > VEL1 or VOL > VOL1, the SSM skips step S35 and passes control to step S36. In the present embodiment, whether the processing of step S35 is to be performed is determined according to the values of velocity VEL and volume VOL. The decision may also be made by detecting the load state of the CPU 1 and according to the detection result, for example.

[0076] In step S36, channel assignment processing based on the note-on event concerned is performed. The channel number of the assigned sound channel is stored in register n allocated at a predetermined location in RAM 3. The contents stored in the register n are hereafter referred to as sound channel n. In step S37, the MIDI-CH voice table shown in FIG. 8 is searched. The timbre data (voice number) of VOICENOp of the part p in the table is converted into a music tone parameter according to the above-mentioned note number NN and velocity VEL. For example, if voice number j is stored in VOICENOp, the music tone parameter VOICEj explained with reference to FIG. 7A is generated. Then, the buffer OPBUFm in each operator data OPmDATAj of the music tone parameter VOICEm is initialized or cleared.

[0077] In step S38, the music tone parameter VOICEj generated in step S37 is transferred or expanded along with time TM into the timbre register TONEPARn corresponding to the sound channel n. At the same time, key-on data KEYONn in the timbre register TONEPARn and each operator-on parameter OPONm are set to "1" (on). Further, in step S39, the computational order is determined among the sound channels assigned for sounding such that music tone generating computations are performed in the order of note-on event occurrence times. To be more specific, the channel numbers are rearranged according to the determined computational order and the rearranged channel numbers are stored in CH sequence register CHSEQ allocated at a predetermined position in the RAM 3, upon which this MIDI processing comes to an end. The CH sequence register CHSEQ is illustrated in FIG. 13.

[0078] In step S40 of FIG. 12, it is determined whether the MIDI event is a note-off event. If the MIDI event is found a note-off event, the SSM passes control to step S41; otherwise, the SSM passes control to step S44. In step S41, the note-off event data concerned is decoded. The note number turned off is stored in the

register NN. At the same time, data indicating the time at which the note-off event should occur is stored in the register TM. In step S42, the sound channel with the note number NN assigned for sounding is searched. The channel number obtained is stored in register i (this value is hereafter referred to as "sound channel i") allocated at a predetermined position in the RAM 3. In step S43, key-off is designated for timbre register TONEPARi corresponding to sound channel i. Namely, after note-off is reserved in the timing corresponding to time TM, this MIDI processing is ended.

[0079] In step S44, it is determined whether the MIDI event is a program change event for changing timbres. If the MIDI event is found a program change event, the data of VOICENOp at the position corresponding to the part p (this part p is not necessarily the part number stored in step S33) designated by the received program change event is changed to value PCHNG designated by the received program change event, upon which this MIDI processing comes to an end (step S45). On the other hand, if the MIDI event is found other than a program change event, the corresponding processing is performed, upon which this MIDI processing comes to an end.

[0080] In this MIDI processing, the timbres corresponding to a plurality of parts are designated in the MIDI-CH voice table. If a note-on event of a plurality of designated parts occurs, a music tone having timbres of the plurality of parts is generated and sounded. Namely, this MIDI processing uses multi-timbre operation specifications. Alternatively, this MIDI processing may use a single-timbre mode in which only a note-on event of a particular part is accepted to generate a music tone of the corresponding timbre.

[0081] FIGS. 14 and 15 are flowcharts indicating detailed procedures of the waveform computation processing subroutine performed in step S17 of FIG. 10. First, a music tone waveform buffer is initialized (step S51). A music tone waveform buffer exists in an area other than a reserved area (buffer) for reproduction in the output buffer. The music tone waveform buffer provides an area for one frame time of waveforms to be generated this time. The initialization of this music tone waveform buffer is to allocate that area in the output buffer and to clear that area. Next, the load state of the CPU 1 is checked (step S52). Based on the check result, a maximum number of channels CHmax that can execute the waveform computation processing is determined (step S53). If the OS always checks the load state of the CPU 1, the check of step S52 may be performed using this load state information. If the OS does not always check the load state of the CPU 1, a routine may be provided that counts a time for looping the main program of FIG. 10 once. The check of step S52 may be performed using a value obtained based on the measured time. Instead of the processing of step S53, processing similar to the processing of step S35 of FIG. 11 may be performed. Namely, the timbre changing process is conducted for changing the timbre assigned to the part to an alternate timbre having a smaller number of constituting operators.

[0082] Then, index i indicating a channel number is initialized to "1 "(step S54). In step S55, the channel number SEQCHNOi stored in SEQCHi at i position in the CH sequence register CHSEQ shown in FIG. 13 is stored in variable n (in this waveform computation processing subroutine, this value is referred to as "channel n"). In step S56, algorithm designation data ALGORn of the music tone register TONEPARn corresponding to channel n is referenced to determine the number of operators (OPs) and the connection mode of each operator to be used in the FM computation processing for channel n.

[0083] Moreover, a computation amount in the current frame is determined according to the note events and the like (step S57). The determination of the computation amount actually denotes determining a net area of the music tone waveform buffer for which the waveform computation processing is to be performed in channel n. The music tone waveform buffer is the area sufficient to store waveform data of one frame time in which the current computation is made. On the other hand, the music tone waveform data of each channel is not necessarily generated all over the area for one frame. Namely, since the sounding timing and muting timing of music tones are different for different channels, a music tone of a certain channel may be turned on or off halfway in the music tone waveform buffer. In view of this, the computation amount must be determined for each channel.

[0084] Next, in step S58 of FIG. 15, the FM computation processing subroutine for generating music tone waveform data for one sample is generated for channel n. In step S59, it is determined whether the music tone generation processing for one frame for channel n has been completed. It should be noted that the determination of step S59 is performed by considering the computation amount determined in step S57. In step S59, if the music tone generation processing for one frame for channel n has not been completed, the SSM passes control back to step S58, in which the music tone waveform data of next sample is generated. If, in step S59, the music tone generation processing for one frame for channel n has been completed, the SSM passes control to step S60.

[0085] In step S60, the music tone waveform data for one frame generated in steps S58 and S59 is written to the music waveform buffer. At this moment, if music tone waveform data is already stored in the music waveform buffer, the data obtained this time is accumulated to the existing data and a result of the addition is written to the music tone waveform buffer. Then, the value of index i is incremented by one (step S61) to determine whether the resultant value of index i is greater than the above-mentioned maximum number of channels CHmax (step S62).

[0086] In step S62, if i ≤ CHmax, or if there are more channels to be processed for the waveform generation, the SSM returns control to step S55, in which the above-mentioned processing operations are repeated. If i > CHmax, or if there is no channel to be processed, muting channel processing for gradually decreasing the size of a volume envelope is performed for the sound channel turned off this time (step S63). In step S64, the music tone waveform data thus generated is removed from the music tone waveform buffer, and the removed data is passed to the CODEC hardware which is an output device. Then, reproduction of the data is instructed, upon which this waveform computation processing comes to an end.

[0087] If the velocity value of channel n gets smaller than a predetermined value, the FM computation for that channel n may not be performed. In order to implement this operation, step S71 is provided after the above-mentioned step S55 as shown in FIG. 14. In step S71, it is determined whether touch velocity data VELn in the timbre register TONEPARn of channel n is higher than predetermined value VELn1. If VELn ≥ VELn1, the SSM passes control to step S56; if VELn < VELn1, key-off is designated for channel n in the similar manner as that of step S43 shown in FIG. 12. Then, the SSM passes control to step S61.

[0088] FIG. 16 is a flowchart indicating the detailed procedure of the FM computation processing subroutine for channel n executed in step S57. Referring to FIG. 16, variable m for storing the operator number of an operator to be processed is initialized (set to "1"). Hereafter, such an operator is referred to as the operator m to be computed. Next, the load state of the CPU 1 is checked and, at the same time, operator priority data OPPRIOm of the operator m to be computed is checked (step S82). Based on the check results, it is determined whether the operator computation processing for the operator m is to be performed (step S83).

[0089] In step S83, if the operator computation processing for the operator m is to be performed, it is determined whether channel n is currently sounding continuously from the preceding frame (step S84). If channel n is found continuously sounding, based on each data stored in the buffer OPBUFm in the operator data OPmDATAn of the timbre register ONEPARn, the operator data OPmDATAn is returned to the state of the operator m at the end of computation of the preceding frame (step S85). The buffer OPBUFm in each operator data OPmDATAn holds the result obtained by the computation performed immediately before. Using this result allows the return to the state of the immediately preceding operator data OPmDATAn. The operator data OPmDATAn is returned to the state at the end of computation of the preceding frame because the music tone waveform data of channel n in the current frame must be generated as the continuation from the preceding frame.

[0090] On the other hand, if channel n is found not sounding continuously from the preceding frame in step S84, the SSM skips step S85 and passes control to step S86. In step S86, the operator computation processing subroutine for the operator m is executed. In step S87, the value of variable m is incremented by one. In step S88, if there are more operators to be processed, the SSM returns control to step S82, in which the above-mentioned processing operations are repeated. If there is no more operator to be processed, the FM computation processing for channel n comes to an end.

[0091] In steps S82 and S83, the load state of the CPU 1 is checked to determine whether the computation of the operator m is to be performed. Alternatively, the computation for the operators having lower priority may not be performed regardless of the load state of the CPU 1. This can increase the number of sound channels when the capacity of the CPU 1 is not so high.

[0092] FIGS. 17 and 18 are flowcharts indicating the detailed procedure of the operator computation processing subroutine for the operator m performed in step S86. FIG. 19 is a diagram illustrating the basic flow of the operator computation to be performed in this operator computation processing. The following explains the operator computation processing for the operator m with reference to FIGS. 17 through 19. Referring to FIG. 17, it is determined whether the operator-on parameter OPONm in the operator data OPmDATAn of the operator m is on ("1") (step S91). If OPONm is "0", or the operator m does not require operator computation, this operator computation processing is ended immediately. If OPONm is "1", or the operator m requires operator computation, the SSM passes control to step S92.

[0093] In step S92, it is determined whether the sampling frequency designation data FSAMPm in the operator data OPmDATAn is "0" or not. Namely, it is determined whether a music tone waveform is to be generated at the sampling frequency FSMAX of standard mode. If FSAMPm = "0", it indicates the standard mode in which each operator performs the music tone waveform generation at the standard sampling frequency. Then, AEGm computation is performed according to the setting value of the envelope parameter EGPARm in the operator data OPmDATAn. The result of this computation is stored in the EG state buffer EGSTATEm (step S93).

[0094] On the other hand, if FSAMPm ≠ "0", for example, FSAMPm = f, the sampling frequency FSMAX of the standard mode is multiplied by 2^{-f} and the music tone waveform generation is performed at the resultant frequency. Namely, in step S94, a parameter of which rate varies (hereafter referred to as a variable-rate parameter) in the envelope parameters EGPARm is multiplied by 2^{f} to perform the AEG computation. The result is stored in the EG state buffer EGSTATEm. The rate of the variable-rate parameter is multiplied by 2^{f} before the envelope generating computation for the following reason. Namely, since the sampling frequency is reduced to FSMAX x 2^{-f}, the time variation of the varia-

ble-rate parameter of the envelope parameter EGPARm is made faster to perform the music tone waveform generation at the sampling frequency concerned. Subsequently, the generated waveform samples are written to 2^f continuous addresses of the buffer, thereby making adjustment such that the resultant music tone has the same pitch as that of the original music tone. Thus, step S93 or S94 performs the computation of envelope data AEGm as shown in FIG. 19.

[0095] In step S95, the data AEGm obtained by the AEGm computation is multiplied by the value of a total level parameter TLm in the operator data OPmDATAn to compute an output level AMPm (= AEGm x TLm) of the operator m as shown in FIG. 19. Then, the amplitude controlling envelope data AEGm computed in step S93 or S94 and the output level AMPm of the operator m computed in step S95 are checked independently (step S96). Based on the check results, it is determined whether the data value AEGm and the data value AMPm are lower than a predetermined time and a predetermined level, respectively, thereby determining in turn whether the operator m is to be operated or not (step S97). In other words, it is determined whether the music tone waveform computation in the operator m may be ended or not. If the decision is YES, the SSM passes control to step S98; if the decision is NO, the SSM passes control to step S101 shown in FIG. 18.

[0096] In step S98, it is determined whether the operator m is a carrier. If the operator m is found a carrier, the SSM passes control to step S99. In step S99, the buffer OPBUF for the operator m and the modulator modulating only the operator m are cleared, the waveform computation is stopped, and this operator computation processing is ended. Thus, if the operator m is a carrier, not only the waveform computation of the operator m but also the waveform computation of the modulator modulating only the operator m is stopped. The carrier is an operator that eventually outputs the music tone waveform data as shown in FIGS. 4A through 4C. If there is no output from the carrier, or if the SSM passes control from step S97 to S99 via S98, it may be assumed that nothing is outputted from the modulator preceding the carrier. If that modulator is modulating another carrier, the waveform computation of that modulator cannot be stopped. On the other hand, if the operator m is found not a carrier in step S98 , or the operator is a modulator, only the buffer OPBUFm of the operator m is cleared to stop the waveform computation (step S100), upon which this operator computation processing comes to an end.

[0097] In step S101 shown in FIG. 18, algorithm designation data ALGORn is checked. In step S102, it is determined whether the operator m is being modulated from another operator. In step S102, if the operator m is found being modulated from another operator, the operator output data stored in the operator output value buffer OPOUTk in each operator data OPkDATAn under modulation are added together, and the result is stored in the data input buffer MODINm of the operator m (step S103). On the other hand, if the operator m is not being modulated by another operator, the SSM passes control to step S104, skipping step S103. In step S104, it is determined whether the sampling frequency designation data FSAMPm in the operator data OPmDATAn is "0". If FSAMPm = "0", the SSM passes control to step S105; if FSAMPm ≠ "0", the SSM passes control to step S110.

[0098] In step S105, a phase value update computation is performed. The updated result is stored in the phase value buffer PHBUFm (the contents thereof hereafter being referred to as phase value PHBUFm) in the operator data OPmDATAn of the operator m. The phase value update computation denotes herein the computation enclosed by dashed line A in FIG. 19. To be more specific, computation MODINm + FBm + FNOn x MULTm + PHBUFm is performed. MODINm and FBm denote the values stored in the modulation data input buffer MODINm in the operator data OPmDATAn and the feedback output value buffer FBm, respectively. FNOn denotes the frequency number FNOn in the music tone parameter VOICEn. MULTm denotes the frequency multiple data MULTm in the operator data OPmDATAn. PHBUFm denotes the last value of the values stored in the phase value buffer PHBUFm in the operator data OPmDATAn.

[0099] In step S106, a table address is computed based on the phase value PHBUFm computed in step S105. From the basic waveform (for example, a waveform selected from among the above-mentioned eight types of basic waveforms) data selected according to the wave select data WSELm of the operator m, data WAVEm (PHBUFm) at the position pointed by this computed address is read. It should be noted that basic waveform data is referred to "basic waveform table." The data WAVEm (PHBUFm) is multiplied by the output level AMPm computed in step S95. The result is stored in the operator output value buffer OPOUTm (= WAVEm (PHBUFm) x AMPm) of the operator m.

[0100] In step S107, feedback sample computation is performed by the following relation, storing the result in the feedback output value buffer FBm of the operator m.

$$0.5 \times (FBm + OPOUTm \times FBLm)$$

OPOUTm denotes the waveform sample data generated in step S106. FBLm denotes the feedback level data FBLm of the parameter m to be computed. The feedback sample computation is performed to prevent parasitic exciter from occurring.

[0101] In step S108, it is determined, as with step S98, whether the operator m is a carrier or not. If the operator m is found a modulator, this operator computation processing is ended immediately. On the other hand, if the operator m is found a carrier, the waveform sample data OPOUTm generated in step S106 is multiplied by the volume data VOLn of the music tone parameter VOICEn. The multiplication result (= OPOUTm x

VOLn) is added to the position indicated by the pointer for pointing the write position of this time in the corresponding waveform buffer. Further, the value of this pointer is incremented by one (step S109), upon which this operator computation processing comes to an end.

[0102] In step S110, phase value update computation is performed, and the result is stored in the phase value buffer PHBUFm. This computation processing in step S110 differs from the computation processing in step S105 only in the added processing indicated by block B in FIG. 19. Since FSAMPm = f ($\neq$ 0), the phase value must be shifted by f bits, or the value of the phase value buffer PHBUFm must be multiplied by 2^f to change the read address of the basic waveform table to that obtained by multiplying the sampling frequency FSMAX by 2^{-f}. Next, likewise step S106, waveform sample generation is performed by the following relation, storing the result in the operator output value buffer OPOUTm.

$$WAVEm\ (2^f \times PHBUFm) \times AMPm$$

[0103] Then, likewise step S107, a feedback sample computation is performed (step S112).

[0104] In step S113, it is determined likewise step S108 whether the operator m is a carrier. If the operator m is found a modulator, this operator computation processing is immediately ended. If the operator m is found a carrier, the waveform sample data OPOUTm generated in step S111 is multiplied by the volume data VOLn of music tone parameter VOICEn. The result (= OPOUTm x VOLn) is added to 2^f addresses of the buffer continued from the position indicated by the pointer in the above-mentioned waveform buffer. Then, the pointer is incremented by 2^f (step S114), upon which this operator computation processing comes to an end. It should be noted that, when writing the plural pieces of sample data of the same value in step S114, interpolation may be made between the pieces of sample data as required, writing the resultant interpolation value to the above-mentioned areas.

[0105] In the present embodiment, as explained in steps S106 and S111, the values stored in the basic waveform table are used for the basic waveform data. Alternatively, the basic waveform data may be generated by computation. Also, the basic waveform data may be generated by combining table data and computation. For the address by which the basic waveform table is read in steps S106 and S110, the address obtained based on the phase value PHBUFm computed in steps S105 and S110 is used. Alternatively, the address obtained by distorting this phase value PHBUFm by computation or by a nonlinear characteristic table may be used.

[0106] FIG. 20 is a flowchart indicating the detailed procedure of the timbre setting processing subroutine of step S19 shown in FIG. 10. Referring to FIG. 20, first, MIDI channels and corresponding timbres are set (step S121). As explained before, in the present embodiment, the MIDI channels and the corresponding timbres are determined from the MIDI-CH voice table. The data to be loaded into this MIDI-CH voice table is stored in the hard disk or the like. When the MIDI file selected by the user is loaded, the corresponding table data is loaded into the MIDI-CH voice table at the same time. Therefore, the processing performed in step S121 is only the editing of the currently loaded data table or the loading of new table data.

[0107] It should be noted that the user may alternatively set the desired number of operators for each of MIDI channels. If the desired number of operators is set to the channel concerned when changing the voice numbers in the MIDI-CH voice table, the voice numbers corresponding to the music tone parameters VOICE equal to or lower than the number of operators may be displayed in a list. From among these voice numbers, the user may select and set desired ones. At this time, the desired number of operators set to the channel concerned may also be automatically changed. The voice numbers within the automatically changed number of operators may be displayed in a list. Moreover, when the user has changed the voice numbers in the MIDI-CH voice table, the total number of operators constituting the music tone parameters VOICE corresponding to the changed voice numbers may be checked. According to the load state of the CPU 1, warning that this timbre cannot be assigned to the channel concerned may be displayed. In addition to such a warning, the voice number of the channel concerned may be automatically changed to the voice number of an alternate timbre obtained by the smaller number of operators.

[0108] As described, the present embodiment is constituted such that the number of operators for use in the FM computation processing can be flexibly changed according to the capacity of the CPU 1, the operating environment of the embodiment, the purpose of use, and the setting of processing. Consequently, the novel constitution can adjust the load of the CPU 1 and the quality of output music tone waveforms without restriction, thereby significantly enhancing the degree of freedom of the sound source system in its entirety. In the present embodiment FM tone generating is used for the music tome waveform generation. It will be apparent that the present invention is also applicable to a sound source that performs predetermined signal processing such as AM (Amplitude Modulation) and PM (Phase Modulation) by combining music tone waveform generating blocks. Further, the CPU load mitigating method according to the invention is also applicable to a sound source based on waveform memory reading and to a physical model sound source in software approach. The present embodiment is an example of personal computer application. It will be apparent that the present invention is also easily applicable to amusement equipment such as game machines, karaoke apparatuses, electronic musical instruments, and general-purpose electronic equipment. Further, the present invention is applicable to a sound source board and a sound source

unit as personal computer options.

[0109] The software associated with the present invention may also be supplied in disk media such as a floppy disk, a magneto-optical disk, and a CD-ROM, or machine-readable media such as a memory card. Further, the software may be added by means of a semiconductor memory chip (typically ROM) which is inserted in a computer unit. Alternatively, the sound source software associated with the present invention may be distributed through the communication interface I/F 11. It may be appropriately determined according to the system configuration or the OS whether the sound source software associated with the present invention is to be handled as application software or device software. The sound source software associated with the present invention or the capabilities of this software may be incorporated in other software; for example, amusement software such as game and karaoke and automatic performance and accompaniment software.

[0110] The inventive machine readable media is used for a processor machine including a CPU and contains program instructions executable by the CPU for causing the processor machine having operators in the form of submodules composed of softwares to compute waveforms for performing operation of generating a plurality of musical tones through a plurality of channels according to performance information. The operation comprises the steps of setting an algorithm which determines a module composed of selective ones of the submodules logically connected to each other to compute a waveform specific to one of the musical tones, designating one of the channels to be used for generating said one musical tone in response to the performance information, loading the selective submodules into said one channel, and logically executing the loaded selective submodules according to the algorithm so as to compute the waveform to thereby generate said one musical tone through said one channel.

[0111] Preferably, the step of setting sets different algorithms which determine different modules corresponding to different timbres of the musical tones. Each of the different modules is composed of selective ones of the submodules which are selectively and sequentially connected to each other to compute a waveform which is specific to a corresponding one of the different timbres.

[0112] Preferably, the step of setting comprises adjusting a number of the submodules to be loaded into the channel dependently on a condition during the course of generating the musical tone.

[0113] Preferably, the step of adjusting comprises compacting the module so as to reduce the number of the submodules when the condition indicates that an amplitude envelope of the waveform attenuates below a predetermined threshold level.

[0114] Preferably, the step of adjusting comprises compacting the module so as to reduce the number of the submodules when the condition indicates that an output volume of the musical tone is tuned below a predetermined threshold level.

[0115] Preferably, the step of adjusting comprises eliminating one submodule so as to reduce the number of the submodules to be loaded into the channel when the condition indicates that said one submodule loses contribution to computation of the waveform without substantially affecting other submodules.

[0116] The inventive machine readable media contains instructions for causing a processor machine having a software module to compute samples of a waveform in response to a sampling frequency for performing operation of generating a musical tone according to performance information. The operation comprises the steps of periodically operating the processor machine to execute the software module based on a variable sampling frequency for successively computing samples of the waveform so as to generate the musical tone, detecting a load of computation imposed on the processor machine during the course of generating the musical tone, and changing the variable sampling frequency according to the detected load to adjust a rate of computation of the samples.

[0117] Preferably, the step of changing provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number.

[0118] FIG. 21 shows a software sound source system practiced as a second preferred embodiment of the present invention. Referring to FIG. 21, a MIDI output section denoted by APS1 is a module for outputting a MIDI message. The APS1 is a performance operator device such as a keyboard, a sequencer for outputting a MIDI message, or application software for outputting a MIDI message. A MIDI API denoted by IF1 is a first application program interface that transfers MIDI messages to an operation system OS. A software sound source module SSM is application software installed in the operating system OS as a driver. The software sound source module SSM receives a MIDI message from the MIDI output section APS1 via the interface IF1. Based on the received MIDI message, the software sound source module SSM generates tone waveform data. The generated tone waveform data is received by the operating system OS via a second application program interface (WAVE out API) IF2 of the OS. An output device OUD is a driver module installed in the operating system OS. The OUD receives the tone waveform data from the software sound source module SSM via the interface IF2, and outputs the received tone waveform data to external CODEC hardware. The output device OUD is software and operates in direct access memory (DMA) manner to read the tone waveform data which is generated by the computation by the software sound source module SSM and stored in a buffer. The OUD supplies the read tone waveform data to the external

hardware composed of a digital-to-analog converter (DAC). The software sound source module SSM includes a tone generator for generating samples of tone waveform data at a predetermined sampling frequency FS by computation, and a MIDI output driver for driving this tone generator. This MIDI output driver reads tone control parameters corresponding to the received MIDI message from a table or the like, and supplies the read parameters to the tone generator.

[0119] FIG. 22 is a timing chart indicating the operation of the software sound source module SSM. As shown, the software sound source module SSM is periodically driven at every frame having a predetermined time length. In computation, the tone control parameters corresponding to the MIDI message received in an immediately preceding frame have been read and stored in a buffer. Based on the various tone parameters stored in the buffer, the SSM generates tone waveform. As shown in FIG. 22, the SSM receives three MIDI messages in a first frame from time T1 to time T2. When computation time T2 comes, the software sound source module SSM is started, upon which the various parameters corresponding to the received MIDI messages are read and stored in the buffer. Based on the received MIDI messages, the SSM performs computation to generate tone waveform data to be newly sounded continuously from the preceding frame.

[0120] In the computation for generating the tone waveform data, a number of samples for one frame is generated for each sound channel. The tone waveform data for all sound channels are accumulated and written to a waveform output buffer. Then, reproduction of the waveform output buffer is reserved for the output device OUD. This reservation is equivalent to outputting of the generated tone waveform data from the software sound source module SSM to the second interface "WAVE out API." The output device OUD reads, for each frame, the tone waveform data a sample by sample from the waveform output buffer reserved for reproduction, and sends the read tone waveform data to the DAC which is the external hardware. For example, from the waveform output buffer which is reserved for reproduction and written with the tone waveform data generated in the first frame from time T1 to time T2, the tone waveform data is read in the second frame from time T2 to Time T3. The read tone waveform data is converted by the DAC into an analog music tone waveform signal to be sounded from a sound system.

[0121] FIG. 23 outlines a processing apparatus having a tone waveform data generator provided by implementing the tone waveform generating method according to the invention. The processor shown in FIG. 23 uses a CPU 1 as the main controller. Under the control of the CPU 1, the tone waveform generating method according to the invention is executed as the tone waveform generation processing based on a software sound source program. At the same time, other application programs are executed in parallel. The CPU 1 is connected, via an internal bus, to a ROM (Read Only Memory) 2, a RAM (Random Access Memory) 3, a display interface 4, an HDD (Hard Disk Drive) 6, a CD-ROM drive 7, an interface 8 for transferring data between the internal bus and an extended bus, and a keyboard 10 which is a personal computer user interface. The CPU 1 is also connected, via the internal bus, the interface 8 and the extended bus, to a digital signal processor (DSP) board 9, a network interface 11, a MIDI interface 12, and a CODEC 14 having a DAC 14-2.

[0122] The ROM 2 stores the operating program and so on. The RAM 3 includes a parameter buffer area for storing various tone control parameters, a waveform output buffer area for storing music tone waveform data generated by computation, an input buffer area for storing a received MIDI message and a reception time thereof, and a work memory area used by the CPU 1. The display 5 and the display interface 4 provide means for the user to interact with the processing apparatus. The HDD 6 stores the operation system OS such as Windows 3.1 (registered trademark) or Windows 95 (registered trademark) of Microsoft Corp., programs for implementing the software sound source module, and other application programs for implementing "MIDI API" and "WAVE API." A CD-ROM 7-1 is loaded in the CD-ROM drive 7 for reading programs and data from the CD-ROM 7-1. The read programs and data are stored in the HDD 6 and so on. In this case, a new sound source program for implementing a software sound source is recorded on the CD-ROM 7-1. The old sound source program can be upgraded with ease by the CD-ROM 7-1 which is a machine readable media containing instructions for causing the personal computer to perform the tone generating operation.

[0123] The digital signal processor board 9 is an extension sound-source board. This board is a hardware sound source such as an FM synthesizer sound source or a wave table sound source. The digital signal processor board 9 is composed of a DSP 9-1 for executing computation and a RAM 9-2 having various buffers and various timbre parameters.

[0124] The network interface 11 connects this processing apparatus to the Internet or the like via a LAN such as Ethernet or via a telephone line, thereby allowing the processing apparatus to receive application software such as sound source programs and data from the network. The MIDI interface 12 transfers MIDI messages between an external MIDI equipment and, receives MIDI events from a performance operator device 13 such as a keyboard instrument. The contents and reception times of the MIDI messages inputted through this MIDI interface 12 are stored in the input buffer area of the RAM 2.

[0125] The CODEC 14 reads the tone waveform data from the waveform output buffer of the RAM 3 in direct memory access manner, and stores the read tone waveform data in a sample buffer 14-1. Further, the CODEC 14 reads samples of the tone waveform data,

one by one, from the sample buffer 14-1 at a predetermined sampling frequency FS (for example, 44.1 kHz), and converts the read samples through a DAC 14-2 into an analog music tone signal, thereby providing a music tone signal output. This tone output is inputted into the sound system for sounding. The above-mentioned constitution is generally the same as that of a personal computer or a workstation. The tone waveform generating method according to the present invention can be practiced by such a machine.

[0126] The following outlines the tone waveform generating method according to the present invention by means of the software sound source module under the control of the CPU-1. When the application program APS1 is started, MIDI messages are supplied to the software sound source module SSM via the first interface IF1. Then, the MIDI output driver of the software sound source module SSM is started to set tone control parameters corresponding to the supplied MIDI messages. These tone control parameters are stored in sound source registers of respective sound channels assigned with the MIDI messages. Consequently, a predetermined number of samples of waveform data are generated by computation in the sound source that is periodically activated every computation frame as shown in FIG. 22.

[0127] FIGS. 24 through 26 show an example of a sound source model based on the tone waveform data generating method according to the present invention. It should be noted that this sound source model is implemented not by hardware but by software. The sound source model illustrated in FIG. 24 through FIG. 26 simulates a wind instrument system or a string instrument system. This model is hereafter referred to as a physical model sound source. The physical model sound source of the wind instrument system simulates an acoustic wind instrument having a mouthpiece at a joint of two tubes as shown in FIG. 24. The physical model sound source of the string instrument system simulates a plucked string instrument or a rubbed string instrument having strings fixed at both ends with bridges.

[0128] The physical model sound source shown in FIG. 24 is composed of a looping circuit. The total delay time in the loop corresponds to a pitch of a music tone to be generated. When the physical model sound source simulates a wind instrument, the sound source includes a circuit for simulating the tube disposed rightward of the mouthpiece. In this circuit, a junction of 4-multiplication grid type composed of four multipliers MU4 through MU7 and two adders AD4 and AD5 simulates a tone hole. Further, a propagation delay in the tube from the mouthpiece to the tone hole is simulated by a delay circuit DELAY-RL. The propagation delay in the tube from the tone hole to the tube end is simulated by a delay circuit DELAY-RR. Acoustic loss of the tube is simulated by a lowpass filter FILTER-R. Reflection at the tube end is simulated by a multiplier MU8. Similarly, in a circuit for simulating the tube disposed leftward of the mouthpiece, the propagation delay of this tube is simulated by a delay circuit DELAY-L. The acoustic loss of the tube is simulated by a lowpass filter FILTER-L. The reflection at the tube end is simulated by a multiplier MU3.

[0129] It should be noted that delay times DRL, DRR, and DL read from a table according to the pitch of the music tone to be generated are set to the delay circuits DELAY-R, DELAY-RR, and DELAY-L, respectively. Filter parameters FRP and FRL for obtaining selected timbres are set to the lowpass filters FILTER-R and FILTER-L, respectively. In order to simulate the acoustic wave propagation mode that varies by opening or closing the tone hole, multiplication coefficients M1 through M4 corresponding to the tone hole open/close operations are supplied to the multipliers MU4 through MU7, respectively. In this case, the pitch of the output tone signal is generally determined by the sum of delay times to be set to the delay circuits DELAY-RL, DELAY-RR, and DELAY L. Since an operational delay time occurs on the lowpass filters FILTER-R and FILTER-L, a net delay time obtained by subtracting this operation delay time is distributively set to the delay circuits DELAY-RL, DELAY-RR, and DELAY-L in a .

[0130] The mouthpiece is simulated by a multiplier MU2 for multiplying a reflection signal coming from the circuit for simulating the right-side tube by multiplication coefficient J2 and a multiplier MU1 for multiplying a reflection signal coming from the circuit for simulating the left-side tube by multiplication coefficient J1. The output signals of the multipliers MU1 and MU2 are added together by an adder AD1, outputting the result to the circuits for simulating the right-side tube and the circuit for simulating the left-side tube. In this case, the reflection signals coming from the tube simulating circuits are subtracted from the output signals by subtractors AD2 and AD3, respectively, the results being supplied to the tube simulating circuits. An exciting signal EX OUT supplied from an exciter and multiplied by coefficient J3 is supplied to the adder D1. An exciter return signal EXT IN is returned to the exciter via an adder AD6. It should be noted that the exciter constitutes a part of the mouthpiece.

[0131] The output from this physical model sound source may be supplied to the outside at any portion of the loop. In the illustrated example, the output signal from the delay circuit DELAY-RR is outputted as an output signal OUT. The outputted signal OUT is inputted into an envelope controller EL shown in FIG. 25, where the signal is attached with an envelope based on envelope parameters EG PAR. These envelope parameters include a key-on attack rate parameter and a key-off release rate parameter. Further, the output from the EL is inputted into a resonator model section RE. The RE attaches resonation formant of the instrument body to the signal based on the supplied resonator parameter. The output signal from the EL is inputted into an effector EF. The EF attaches a desired effect to a music signal

TONE OUT based on supplied effect parameters. The EF is provided for attaching various effects such as reverberation, chorus, delay, and pan. The music tone signal TONE OUT is provided in the form of samples of tone waveform data at every predetermined sampling period.

[0132] FIG. 26 shows an example of the exciter that constitutes a part of the mouthpiece. The exciter return signal EX IN is supplied to a subtractor AD11 as a signal equivalent to the pressure of an air vibration wave to be fed back to the reed in the mouthpiece. From this signal, a blowing pressure signal P is subtracted. The output from the subtractor AD11 provides a signal equivalent to the pressure inside the mouthpiece. This signal is inputted into an exciter filter FIL10 simulating the response characteristics of the reed relating to pressure change inside the mouthpiece. At the same time, this signal is inputted into a nonlinear converter 2 (NLC2) simulating saturation characteristics of the velocity of the air flow inside the mouthpiece relating to the air pressure inside the mouthpiece when gain adjustment is performed by a multiplier MU11. A cutoff frequency of the exciter filter FIL10 is controlled selectivity by a supplied filter parameter EF. The output signal from the exciter filter FIL10 is adjusted in gain by a multiplier MU10. The adjusted signal is added with an embouchure signal E equivalent to the mouthing pressure of the mouthpiece by an adder AD10, providing a signal equivalent to the pressure applied to the reed. The output signal from the adder AD10 is supplied to the nonlinear converter (NLC1) simulating the reed open/close characteristics. The output of the nonlinear converter 1 and the output of the nonlinear converter 2 are multiplied with each other by a multiplier MU12, from which a signal equivalent to the volume velocity of the air flow passing the gap between the mouthpiece and the reed is outputted. The signal outputted from the multiplier MU12 is adjusted in gain by a multiplier MU13,and is outputted as the exciting signal EX OUT.

[0133] The source model simulating a wind instrument has been explained above. In simulating a string instrument, a circuit for simulating a rubbed string section or a plucked string section in which a vibration is applied to a string is used instead of the circuit for simulating the mouthpiece. Namely, the signal P becomes an exciting signal corresponding to a string plucking force and a bow velocity, and the signal E becomes a signal equivalent to a bow pressure. It should be noted that, in simulating a string instrument, a multiplication coefficient NL2G supplied to the multiplier MU11 is made almost zero. Further, by setting the output of the nonlinear converter 2 to a predetermined fixed value (for example, one), the capability of the nonlinear converter 2 is not used. The delay circuits DELAY-RL, DELAY-RR, and DELAY-L become to simulate string propagation times. The lowpass filters FILTER-R and FILTER-L become to simulate string propagation losses. In the exciter, setting of the multiplication coefficients NLG1, NLG2, NL1, and NL2 allows the exciter to be formed according to a model instrument to be simulated.

[0134] The following explains various data expanded in the RAM 3 with reference to FIG. 27. As described above, when the software sound source module SSM is started, the MIDI output driver therein is activated, upon which various tone control parameters are stored in the RAM according to the inputted MIDI messages. Especially, if the MIDI messages designate a physical model sound source (also referred to as a VA sound source) as shown in FIGS. 24 through 26, a tone control parameter VATONEPAR for the selected VA sound source is stored in the control parameter buffer (VATONEBUF) arranged in the RAM 3. The tone waveform data generated by computation by the software sound source module SSM for every frame is stored in the waveform output buffer (WAVEBUF) in the RAM 3. Further, the contents of each MIDI message inputted via the interface MIDI API and the event time of reception of the inputted message are stored in MIDI input buffers (MIDI RCV BUF and TM) in the RAM 3. Further, the RAM 3 has a CPU work area.

[0135] The buffer VATONEBUF stores the tone control parameter VATONEPAR as shown in FIG. 28. The VATONEBUF also stores a parameter SAMPFREQ indicating an operation sampling frequency at which samples of the tone waveform data are generated, a key-on flag VAKEYON which is set when a key-on event contained in a MIDI message designates the VA sound source, a parameter PITCH(VAKC) for designating a pitch, a parameter VAVEL for designating a velocity when the key-on event designates the VA sound source, and a breath controller operation amount parameter BRETH CONT. Moreover, the VATONEBUF has a pressure buffer PBUF for storing breath pressure and bow velocity, a PBBUF for storing a pitch bend parameter, an embouchure buffer EMBBUF for storing an embouchure signal or a bow pressure signal, a flag VAKONTRUNCATE for designating sounding truncate in the VA sound source, and a buffer miscbuf for storing volume and other parameters.

[0136] The parameter SAMPFREQ can be set to one of two sampling frequencies, for example. The first sampling frequency is 44.1 kHz and the second sampling frequency is a half of the first sampling frequency, namely 22.05 kHz. Alternatively, the second sampling frequency may be double the first sampling frequency, namely 88.2 kHz. These sampling frequencies are illustrative only, hence not limiting the sampling frequencies available in the present invention. Meanwhile, if the sampling frequency is reduced 1/2 times FS, the number of the tone waveform samples generated in one frame may be reduced by half. Consequently, if the load of the CPU 1 is found heavy, the sampling frequency of 1/2 times FS may be selected to mitigate the load of the CPU 1, thereby preventing dropping of samples from generation.

[0137] If the sampling frequency is set to 2 times

FS, the number tone waveform samples generated is doubled, allowing the generation of high-precision tone waveform data. Consequently, if the load of the CPU 1 is found light, the sampling frequency of 2 times FS may be selected to generate samples having high-precision tone waveform data. For example, let the standard sampling frequency in the present embodiment be FS1, a variation sampling frequency FS2 is represented by:

FS1 = n times FS2 (n being an integer) ... first example,
FS1 = 1/n times FS2 (n being an integer) ... second example.

Because the present invention mainly uses the first example, the following description will be made mainly with reference to the first example.

[0138] In the present invention, the sampling frequencies of the tone waveform data to be generated are variable. If there is another acoustic signal to be reproduced by the CODEC, the sampling frequency of the DA converter in the CODEC may be fixed to a particular standard value. For example, when mixing the music tone generated by the software sound source according to the present invention with the digital music tone outputted from a music CD, the sampling frequency may be fixed to FS1 = 44.1 kHz according to the standard of the CD. The following explains an example in which the sampling frequency of the CODEC is fixed to a standard value. The relation between this standard sampling frequency FS1 and the variation sampling frequency FS2 is represented by FS1 = n times FS2 as described before. The sampling frequency of the DA converter is fixed to the standard value. Therefore, it is required for the waveform output buffer WAVEBUF which is read a sample by sample every period of this fixed standard sampling frequency FS1 to store beforehand a series of the waveform data in matching with the standard sampling frequency FS1 regardless of the sampling frequency selected for the waveform computation. If the sampling frequency FS2 which is 1/n of the sampling frequency FS1 is selected, the resultant computed waveform samples are written to the waveform output buffer WAVEBUF such that n samples of the same value are arranged on continuous buffer addresses. When the waveform data for one frame has been written to the waveform output buffer WAVEBUF, the contents of the waveform buffer WAVEBUF may be passed to the CODEC. Since the sampling frequency FSb of the data series stored in the waveform output buffer WAVEBUF differs from the operation sampling frequency FSc of the CODEC (or DAC), sampling frequency matching may be required. For example, if FSb = k times FSc (K > 1), then the tone waveform data may be sequentially passed from the waveform output buffer WAVEBUF in skipped read manner by updating every n addresses. Namely, during the time from the processing of storing the music waveform samples in the waveform output buffer WAVEBUF to the processing of the DAC of the CODEC, a sampling frequency conversion circuit may be inserted to match the write and read sampling frequencies.

[0139] Information about the time at which storage is made in the MIDI event time buffer TM is required for performing the time-sequential processing corresponding to occurrence of note events. If the frame time is set to a sufficiently short value such as 5 ms or 2.5 ms, adjustive fine timing control for various event processing operations is not required substantially in the frame , so that these event processing operations need not be performed by especially considering the time information. However, it is preferable that the information from the breath controller and so on be handled on a last-in first-out basis, so that, for the event of this information, processing on the last-in first-out basis is performed by use of the time information. In addition to the above-mentioned buffers, the RAM 3 may store application programs.

[0140] FIG. 28 shows details of the tone control parameters VATONEPAR. The tone control parameters VATONPAR include an exciter parameter (EXCITER PARAMETERS), a wind instrument/string instrument parameter (P/S PARAMETERS), an envelope parameter (EG PAR), a resonator parameter (RESONATOR PAR), an effect parameter (EFFECT PAR), and sampling frequency data (SAMPLING FREQ). Each of these parameters includes a plurality of parameter items. Each delay amount parameter and each tone hole junction multiplication coefficient are determined by a pitch of a musical tone. In this case, DL through DRR are tables listing a delay amount for a pitch. Delay amounts are read from these tables and set so that a total delay amount corresponds to a desired pitch. Each of these delay amount tables is prepared by actually sounding a tone having a predetermined pitch and by feeding back a deviation in the pitch frequency. The filter parameters such as FLP and FRP are set according to the contour of the tube to be simulated, the characteristics of the string, and the operation amount of the operator device. It should be noted that preferred tone control parameters VATONEPAR are set according to the sampling frequency used. The sampling frequency of these tone control parameters VATONEPAR is indicated by SAMPLING FREQ in FIG. 28. The processing for waveform generation by computation is performed by using the tone control parameters VATONEPAR prepared for the sampling frequency concerned by referencing this SAMPLING FREQ information. In this example, the standard sampling frequency is FS1 and the alternative sampling frequency FS2 is 1/2 times FS1, for example.

[0141] As descrived above, the inventive sound source apparatus has a software module used to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information. In the inventive apparatus, a

processor device composed of the CPU 1 periodically executes the software module SSM for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone. A detector device included in the CPU 1 detects a load of computation imposed on the processor device during the course of generating the musical tone. A controller device implemented by the CPU 1 operates according to the detected load for changing the variable sampling frequency to adjust a rate of computation of the samples.

[0142] Preferably, the controller device provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number.

[0143] The processor device includes a delay device having a memory for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay device generates a write pointer for successively writing the samples into addresses of the memory and a read pointer for successively reading the samples from adresses of the memory to thereby create the delay corresponding to an address gap between the write pointer and the read pointer. The delay device is responsive to the fast sampling frequency to increment both of the write pointer and the read pointer by one address for one sample. Otherwise, the delay device is responsive to the slow sampling frequency to increment the write pointer by one address n times for one sample and to increment the read pointer by n addresses for one sample.

[0144] The processor device may include a delay device having a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay device successively writes the samples of the waveform of one mucical tone into addresses of one of the memory regions, and successively reads the samples from addresses of the same memory region to thereby create the delay. The delay device is operative when said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from addresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

[0145] Preferably, the processor device executes the software module composed of a plurality sub-modules for successively computing the waveform. The processor device is operative when one of the sub-modules declines to become inactive without substantially affecting other sub-modules during computation of the waveform for skipping execution of said one sub-module.

[0146] The inventine sound source apparatus has a software module used to compute samples of a waveform for generating a musical tone. In the inventive apparatus, a provider device variably provides a trigger signal at a relatively slow rate to define a frame period between successive trigger signals, and periodically provides a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period. The processor device is resettable in response to each trigger signal and is operable based on each sampling signal to periodically execute the software module for successively computing a number of samples of the waveform within one frame. The detector device detects a load of computation imposed on the processor device during the course of generating the musical tone. The controller device is operative according to the detected load for varying the frame period to adjust the number of the samples computed within one frame period. A converter device composed of CODEC 14 is responsive to each sampling signal for converting each of the samples into a corresponding analog signal to thereby generate the musical tones.

[0147] The following explains the operations of the present invention in detail with reference to flowcharts.

[0148] FIG. 29 is a flowchart showing an initialization program to be executed at a power-on or reset sequence. When the initialization program is started, system initialization such as hardware initialization is performed in step SS10. Next, in step SS11, the OS program is started to place other programs in an executable state in which a main program for example is executed.

[0149] FIG. 30 is a flowchart showing the main program to be executed by the CPU 1. When the main program is started, initialization such as resetting of registers is performed in step SS20. Next, in step SS21, basic display processing such as arranging windows for display screens such as desktop is performed. Then, in step SS22, trigger check is performed for task switching. In step SS23, it is determined whether a trigger has taken place. The operations of steps SS21 through SS23 are repeated cyclically until a trigger is detected. If a trigger is found, the decision in step SS23 turns YES. In step SS24, the task switching is performed so that a task corresponding to the detected trigger is executed.

[0150] There are five types of triggers for commencing the task switching. If supply of a MIDI message from an application program or the like via the sound source API (MIDI API) is detected, it indicates trigger 1. In this case, the software sound source module SSM is started in step SS25 to perform MIDI processing. If an internal interrupt has been caused by a software timer (tim) that outputs the interrupt every frame period, it indicates trigger 2. In this case, the software sound source module SSM is started in step SS26 to perform waveform computation processing, thereby generating tone waveform data for the predetermined number of

samples. If a transfer request for tone waveform data has been made by an output device (CODEC) based on DAM, it indicates trigger 3. In this case, transfer processing is performed in step SS27 in which the tone waveform data is transferred from the waveform output buffer WAVEBUF to the output device. If an operation event based on manual operation of the input operator device such as the mouse or the keyboard of the processing apparatus has been detected, it indicates trigger 4. In the case of the operation event for timbre setting, timbre setting processing is performed in step SS28. For other operation events, corresponding processings are performed in step SS29. If the end of the operation has been detected, it indicates trigger 5. In this case, end processing is performed in step S30. If no trigger has been detected, trigger 4 is assumed and the processing of steps SS28 and SS29 is performed. When the processing of trigger 1 to trigger 5 has been completed, the SSM returns control to step SS21. The processing operations of steps SS21 through SS30 are repeated cyclically.

[0151] FIG. 31 is a flowchart showing the MIDI processing to be performed in step SS25. When the MIDI processing is started, the contents of the MIDI event are check in step S40. This check is specifically performed on the MIDI message written to "MIDI API" constituted as a buffer. Then, it is determined in step SS41 whether the MIDI event is a note-on event. If the MIDI event is found a note-on event, the SSM passes control to step SS42, in which it is determined whether the sound channel (MIDI CH) assigned to that note-on event belongs to a physical model sound source or a VA sound source. If the sound channel assigned to the note-on event is found in the physical model sound source(hereafter, such a MIDI CH is labeled "VA CH"), the key-on processing in the physical model sound source is performed in step SS43 and control is returned. If the sound channel assigned to the note-on event is not found in the physical model sound source, the key-on processing of another sound source is performed in step SS44, upon which control is returned. This key-on processing is performed in the DSP 9-1 of the digital signal processing board 9, for example.

[0152] If the MIDI event is found not a note-on event in step SS41, it is determined in step SS45 whether the MIDI event is a note-off event. If the MIDI event is found a note-off event, it is determined in step SS46 whether the sound channel (MIDI CH) assigned to the note-off event belongs to the physical model sound source. If the sound channel assigned to the note-off event is found in the physical model sound source, the key-on flag VAKEYON in the physical model sound source is set to "0" in step SS47, and the occurrence time of the note-off event is stored in the MIDI event time buffer TM, upon which control is returned. If the sound channel assigned to the note-off event is not found in the physical model sound source, the key-off processing of another sound source is performed in step SS48, upon which control is returned.

[0153] Further, if the MIDI event is found not a key-off event in step SS45, it is determined in step SS 49 whether the MIDI event is a program change. If the MIDI event is found the program change, it is determined in step SS50 whether the sound channel (MIDI CH) assigned to the MIDI event of program change belongs to the physical model sound source. If the sound channel assigned to the MIDI event of program change is found in the physical model sound source, the tone control parameters VATONEPAR designated in the program change are stored in step SS51; upon which control is returned. If the sound channel assigned to the MIDI event of program change is not found in the physical model sound source, the timbre parameter processing corresponding to that sound channel is performed in step SS52, upon which control is returned. If the MIDI event is not a program change in step SS49, the processing of the corresponding MIDI event is performed in step SS53, upon which control is returned. In this MIDI event processing, the processing for a breath controller operation is performed, for example.

[0154] FIG. 32A is a flowchart showing the key-on processing of the physical model sound source to be performed in step SS43. When the physical model sound source key-on processing is started, the note number contained in the received MIDI message is stored in the buffer VATONEBUF as a parameter VAKC in step SS55. The velocity information contained in the same MIDI message is stored in the VATONEBUF as a parameter VAVEL. The VAKEYON flag is set to "1". Further, the MIDI message receive time is stored in the buffer TM as an event occurrence time. Pitch frequency data converted from the parameter VAKC and the pitch bend value stored in the pitch bend buffer PBBUF are stored in the buffer VATONEBUF as a parameter PITCH. When these processing operations come to an end, control is returned. It should be noted that, instead of using the pitch bend value for obtaining the pitch frequency, the pitch bend value may be used for setting an embouchure parameter.

[0155] FIG. 32B is a flowchart showing the timbre setting processing to be performed in step SS28 when the above-mentioned trigger 4 has been detected. When the user performs a timbre setting operation by manipulating the mouse or keyboard, the timbre setting processing is started. In step SS50, it is determined whether timbre setting of the physical model sound source has been designated. If the timbre setting is found designated, the timbre parameter corresponding to the designated timbre is expanded in the buffer VATONEBUF as shown in FIG. 27 in step SS61. Then, in step SS62, the timbre parameter is edited by the user, upon which the timbre setting processing comes to an end. If the timbre setting is found not designated in step SS60, control is passed to step SS62, in which the timbre parameter is edited by the user and the timbre setting processing comes to an end.

[0156] FIG. 32C is a flowchart showing other MIDI event processing to be performed in step SS53. When the other MIDI event processing is started, it is determined in step SS65 whether the sound channel (MIDI CH) assigned to the MIDI event belongs to the physical model sound source. If the sound channel assigned to the MIDI event is found in the physical model sound source, it is determined in step SS66 whether the MIDI event is a breath control event. If the MIDI event is found a breath control event, the parameter BRETH CONT in the breath control event is stored in the pressure buffer PBUF in step SS67.

[0157] If the MIDI event is found not a breath control event, step SS67 is skipped, and, in step SS68, it is determined whether the MIDI event is a pitch bend event. If the MIDI event is found a pitch bend event, it is determined in step SS69 whether the embouchure mode is set. If the embouchure mode is set, the parameter PITCHBEND in the pitch bend event is stored in the embouchure buffer EMBBUF in step SS70. If the embouchure mode is not set, the parameter PITCHBEND in the pitch bend event is stored in the pitch bend buffer PBBUF in step SS72.

[0158] Further, if it is found that the sound channel does not belong to the physical model sound source in step SS65 and if the MIDI event is found not a pitch bend event in step SS68, control is passed to step SS71, in which it is assumed that the received MIDI event does not correspond to any of the above-mentioned events, then processing corresponding to the received event is performed, and control is returned. It should be noted that the embouchure signal indicates a pressure with which the player mouths the mouthpiece. Since the pitch varies based on this embouchure signal, the parameter PITCHBEND is stored in the embouchure buffer EMBBUF in the embouchure mode. As described, every time a MIDI event is received, the parameters associated with music performance are updated by the MIDI event processing.

[0159] FIG. 33 is a flowchart showing the physical model parameter expanding processing. This processing is performed in step SS61 of the above-mentioned timbre setting processing before sounding. When the physical model parameter expanding processing is started, the CPU load state is checked in step SS75. This check is performed based on a status report from the CPU 1 for example and by considering the setting value of the sampling frequency FS. If this check indicates in step SS76 that the load of the CPU 1 is not yet heavy, the shortest frame period of one frame set by the user or the standard frame period TIMDEF is set in step SS77 as a period tim of the software timer that causes a timer interrupt for conducting the waveform generation processing every frame. It should be noted that the standard frame period TIMDEF is set to 2.5ms, for example.

[0160] In step SS78, the sampling frequency FS specified by the tone control parameter VATONEPAR for the selected physical model sound source is set as the operation sampling frequency SAMPFREQ. Further, in step SS79, alarm clear processing is performed. In step SS80, the tone control parameters VATONEPAR containing to the parameter SAMPFREQ and the parameter VAKC are read to be stored in the buffer VAPARBUF, upon which control is returned. In this case, the tone control parameters VATONEPAR considering the parameter VAVEL may be stored in the buffer VAPARBUF.

[0161] If the load of the CPU 1 is found heavy in step SS76, it is determined in step SS81 whether the frame time automatic change mode is set. If this mode is set, a value obtained by multiplying the standard frame period TIMDEF by integer α is set as the period tim of the software timer in step SS82. Integer α is set to a value higher than one. When the frame period is extended, the frequency at which parameters are loaded into the physical model sound source can be lowered, thereby reducing the number of processing operations for transferring the changed data and the number of computational operations involved in the data updating.

[0162] In step SS83, the current operation sampling frequency SAMPFREQ is checked. If the operation sampling frequency SAMPFREQ is the sampling frequency FS1, it indicates that the load of the CPU 1 is heavy, so that the sampling frequency FS2 which is 1/2 of FS1 is set as the operation sampling frequency SAMPFREQ in step SS84. Then, the processing operations of step SS79 and subsequent steps are performed. In this case, a new tone control parameter VATONEPAR corresponding to the changed parameter SAMPFREQ is read and stored in the buffer VAPARBUF.

[0163] In step SS83, if the operation sampling frequency SAMPFREQ is found not the standard sampling frequency FS1, alarm display processing is performed in step SS85. This is because the current operation sampling frequency SAMPFREQ is already 1/n times FS1. Although the sampling frequency FS2 that should comparatively reduce the load of the CPU 1 is already set, the load of the CPU 1 has been found heavy. This may disable the normal waveform generation processing in the physical model sound source. If the physical model sound source is found sounding in step SS86, the physical model sound source is muted and the processing of step SS80 is performed.

[0164] The above-mentioned processing operations cause the tone control parameters VATONEPAR necessary for the physical model sound source to generate the waveform data which are stored in the buffer VAPARBUF. This allows the generation of waveforms by computation. In this waveform generation processing, the operation sampling frequency is dynamically changed depending on the load of the CPU 1. Flowcharts for this waveform generation processing of the physical model sound source are shown in FIGS. 34

and 35. The waveform generation processing is started by the timer interrupt outputted from the software timer in which the period tim is set. In step SS90, it is determined whether the key-on flag VAKEYON is set to "1". If the key-on flag VAKEYON is found "1", a computation amount necessary for one frame is computed in step SS91. This computation amount includes the number of samples for generating a continued tone. If the MIDI message received in an immediately preceding frame includes a key-on event, this computation amount includes those for generating the number of samples of a tone to be newly sounded. The number of samples of the tone to be newly sounded may be the number of samples necessary during the time from reception of the MIDI message to the end of the frame concerned.

[0165] Then, in step SS92, the load state of the CPU 1 is checked. This check is performed by considering the occupation ratio of the waveform computation time in one frame period in the preceding frame. If this check indicates in step SS93 that the load of the CPU 1 is not heavy, the sampling frequency FS in the selected tone control parameters VATONEPAR is set as the operation sampling frequency SAMPFREQ in step SS94. If the check indicates that the load of the CPU 1 is heavy, it is determined in step SS105 whether the operation sampling frequency SAMPFREQ can be lowered. If it is found that the operation sampling frequency SAMPFREQ can be lowered, the same is actually lowered in step SS106 to 1/n, providing the sampling frequency FS2. If the sampling frequency is already FS2, and therefore the operation sampling frequency SAMPFREQ cannot be lowered any more, alarm display is performed in step SS107. This is because the operation sampling frequency SAMPFREQ is already set to 1/n times FS1. Although the sampling frequency is already set to the sampling frequency FS2 that should comparatively lower the load of the CPU 1, the actual load of the CPU 1 is found yet heavy. In this case, the necessary computation amount cannot be provided in one frame time or a predetermined time. Then, if the physical model sound source is found sounding in step SS108, the sound channel is muted, upon which control is returned.

[0166] When the processing of step SS94 or step SS106 comes to an end, alarm clear processing is performed in step SS95. Then, in step SS96, it is determined whether the operation sampling frequency SAMPFREQ has been changed. If the operation sampling frequency SAMPFREQ is found changed, the parameter change processing due to the operation sampling frequency change is performed in step SS97. Namely, the tone control parameter VATONEPAR corresponding to the operation sampling frequency SAMPFREQ is read and stored in the buffer VAPARBUF. If the change processing is found not performed, step SS97 is skipped.

[0167] In step SS98, it is determined whether truncate processing is to be performed. This truncate processing is provided for monotone specifications. In the truncate processing, a tone being sounded is muted and a following tone is started. If a truncate flag VATRUNCATE is set to "1", the decision is YES and the truncate processing is started. Namely, in step SS99, the signal P for breath pressure or bow velocity and the signal E for embouchure or bow pressure are set to "0". In step SS100, envelope dump processing is performed. This dump processing is performed by controlling the EG PAR to be supplied to the envelope controller. In step SS101, it is determined whether the envelope dump processing has ended. If this dump processing is found ended, the delay amount set to the delay circuit in the loop is set to "0" in step SS102. This terminates the processing for muting the sounding tone.

[0168] Then, in step SS109 shown in FIG. 35, the data stored in the pressure buffer PBUF is set as a signal P. The data stored in the embouchure buffer EMBBUF is set as a signal E. Further, the frequency data converted based on the key code parameter VAKC and the pitch bend parameter stored in the pitch bend buffer PBBUF is set as a pitch parameter PITCH. Instep SS110, based on the tone control parameters VATONEPAR stored in the buffer VAPARBUF, physical model computation processing is performed. Every time this computation processing is performed, the tone waveform data for one sample is generated. The generated tone waveform data is stored in the waveform output buffer WAVEBUF.

[0169] In step SS111, it is determined whether the waveform computation for the number of samples calculated in step SS91 has ended. If the computation is found not ended, control is passed to step SS113, in which the time occupied by computation by the CPU 1 in one frame time or a predetermined time is checked. If this check indicates that the occupation time does not exceed the one frame time, next sample computation processing is performed in step SS110. The processing operations of steps SS110, SS111, SS113, and SS114 are cyclically performed until the predetermined number of samples is obtained as long as the occupation time does not exceed the one frame time. Consequently, it is determined in step SS111 that the computation of the predetermined number of samples in one frame has ended. Then, in step SS112, the tone waveform data stored in the waveform output buffer WAVEBUF is passed to the output device (the CODEC).

[0170] If it is determined in step SS114 that one frame time has lapsed before the predetermined number of samples has been computed, then, in step SS115, the muting processing of the tone waveform data in the waveform output buffer WAVEBUF is performed. Next, in step SS112, the tone waveform data stored in the waveform output buffer WAVEBUF is passed to the output device (the CODEC). If, in step SS90, the key-on flag VAKEYON is found not to set "1", it is determined in step SS103 whether key-off processing is on. If the decision is YES, the key-off processing

is performed in step SS104. If the key-off processing is found not on, control is returned immediately.

[0171] According to the invention, the tone generating method uses a hardware processor in the form of the CPU 1 and a software module in the form of the sound source module SSM to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information. The inventive method comprises the steps of periodically operating the hardware processor to execute the software module for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone, detecting a load of computation imposed on the hardware processor during the course of generating the musical tone, and changing the variable sampling frequency according to the detected load to adjust a rate of computation of the samples. Preferably, the step of changing provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number.

[0172] The inventive method uses a hardware processor having a software module used to compute samples of a waveform for generating a musical tone. The inventive method comprises the steps of variably providing a trigger signal at a relatively slow rate to define a frame period between successive trigger signals, periodically providing a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period, operating the hardware processor resettable in response to each trigger signal and operable in response to each sampling signal to periodically execute the software module for successively computing a number of samples of the waveform within one frame, detecting a load of computation imposed on the software processor during the course of generating the musical tone, varying the frame period according to the detected load to adjust the number of the samples computed within one frame period, and converting each of the samples into a corresponding analog signal in response to each sampling signal to thereby generate the musical tones.

[0173] Meanwhile, in order to build the physical model sound source in which the sampling frequency is variable, a delay device is required in which the sampling frequency is variable while a delay time can be set without restriction from the sampling frequency. The following explains such a delay device with reference to FIG. 38. In the physical model sound source, each delay circuit uses a delay area in the RAM 3 as a shift register to obtain a predetermined delay amount. A DELAYx 20 shown in FIG. 38 is the delay circuit constituted by the delay area allocated in the RAM 3. The integer part of the delay amount provides the number of shift register stages D between a write pointer indicating an address location at which inputted data is written and a read pointer indicating an address location at which the data is read. The decimal fraction of the delay amount provides multiplication coefficient d to be set to a multiplier MU21 to perform interpolation between a pair of the data read at an address location indicated by the read pointer and the data read at an address location (READ POINTER-n) n stages before that read pointer. It should be noted that a multiplication coefficient (1 - d) is set to a multiplier MU20 for interpolation.

[0174] In this case, a total delay amount of the delay outputs of an adder AD20 in the DELAYx 20 becomes (D + d) equivalent to the number of delay stages. In the equivalent of time, the total delay amount becomes (D + d)/FS for the sampling frequency FS. If the maximum value among the sampling frequencies is FS1, then it is desired to constitute the delay such that the periodic time of the sampling frequency FS1 basically corresponds to one stage of the delay circuit. In such a constitution, in order to lower the sampling frequency to 1/n of the FS1, one sample obtained by the computation may be written to n continuous stages of the delay circuit at n continuous addresses for each sample computation. On the other hand, the delay outputs may be read by updating the read pointer by n addresses. Therefore, in the above-mentioned constitution, the equivalent value of the number of delay stages (D + d) for implementing necessary delay time Td is (D + d) = Td times FS1 regardless of the sampling frequency. It should be noted that the write pointer and the read pointer are adapted to equivalently shift in the address direction indicated by arrow on the shift register. When the pointers reach the right end of the shift register, the pointers jump to the left end, thus circulating on the DELAYx 20.

[0175] As described, since the delay time length of the time equivalent of one stage of delay is made constant (1/FS1) regardless of the sampling frequency FS, the write pointer is set to write one sample of the waveform data over continuous n addresses to maintain the delay time length of the delay output even if the sampling frequency FS is changed to the sampling frequency FS2 which is 1/n of FS1. Every time one sample of the waveform data is generated, the write pointer is incremented by n addresses. The read pointer is updated in units of n addresses (n -1) at once to read the sample delayed by address skipping. This constitution allows the delay output the one sample of the generated waveform data to correspond to the delay output read from the address location before n addresses. Therefore, for the decimal fraction delay part shown in FIG. 38, data before one sample for interpolation is read from an address location n stages (n addresses) before the read pointer.

[0176] Also, in a unit delay means provided for a filter and so on in the physical model sound source, a means generally similar to the above-mentioned delay circuit is used to prevent the delay time length from

being changed even if the preset sampling frequency is changed. The following explains this unit delay means with reference to FIG. 39. The unit delay means also uses the delay area in the RAM 3 as a shift register. A DELAYx 21 shown in FIG. 39 is the unit delay means composed of the delay area allocated in the RAM 3. The unit delay amount of this means is obtained by the shift register through n stages between an address location indicated by a write pointer to which data is written and an address location indicated by a read pointer from which data is read.

[0177] As described with the delay circuit shown in FIG. 38, one sample is written into n consecutive addresses (n stages). Therefore, the address difference between the write pointer and the read pointer is n addresses. In this case, the write pointer is set such that the same value of one sample is written over n addresses. The read pointer is set such that data is read by updating the read pointer in units of n addresses. It should be noted that the unit delay means, by nature, may be constituted only by n stages of delay areas.

[0178] The inventive sound source apparatus has a software module used to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information. In the inventive apparatus, a processor device responds to a variable sampling frequency to periodically execute the software module for successively computing samples of the waveform so as to generate the musical tone. A detector device detects a load of computation imposed on the processor device during the course of generating the musical tone. A controller device operates according to the detected load for changing the variable sampling frequency to adjust a rate of computation of the samples. The controller device provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number. The processor device includes a delay device having a memory for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay device generates a write pointer for successively writing the samples into addresses of the memory and a read pointer for successively reading the samples from adresses of the memory to thereby create the delay corresponding to an address gap between the write pointer and the read pointer. The delay device is responsive to the fast sampling frequency to increment both of the write pointer and the read pointer by one address for one sample. Otherwise, the delay device is responsive to the slow sampling frequency to increment the write pointer by one address n times for one sample and to increment the read pointer by n addresses for one sample.

[0179] The reproduction sampling frequency of the CODEC 14 is generally fixed as described before. If the sampling frequency of the waveform data generated by computation is changed to 1/n, one sample of the generated tone waveform data is repeatedly written, in units of n pieces, to the continuous address locations in the waveform output buffer of the RAM 3. Consequently, in the present embodiment, a series of the waveform data for one frame is written into the waveform output buffer WAVEBUF in the manner corresponding to the sampling frequency FS1. The CODEC 14 operates at the sampling frequency FS1. The CODEC 14 may receive the contents of the waveform output buffer WAVEBUF without change, and may perform DA conversion on the received contents at the sampling frequency FS1. If the reproduction sampling frequency of the CODEC 14 is synchronously varied with the sampling frequency of the waveform data to be generated, the generated waveform data may be written, a sample by sample, to the waveform output buffer WAVEBUF in the RAM 3.

[0180] In the waveform generation processing shown in FIGS. 34 and 35, the tone control parameter VATONEPAR adapted to the sampling frequency FS is read and stored in the buffer VAPARBUF as a parameter to be used for generating tone waveform data. Hence, the tone control parameters VATONEPAR of various timbres are stored in a storage means for each possible sampling frequency FS. An example of the arrangement of these parameters is shown in FIG. 40A. In this example, VATONEPAR1(FS1) and VATONEPAR1(FS2) are tone control parameters for piano. VATONEPARk(FS1) and VATONEPARk(FS2) are tone control parameters for violin. Thus, the tone control parameters having voice numbers VATONEPAR1 through VATONEPARk are a set of parameters prepared for each sampling frequency. The tone control parameters having voice numbers subsequent to VATONEPAR(K+1) provide separate timbres, and correspond to one of the sampling frequency FS1 and the sampling frequency FS2.

[0181] Another example of the arrangement of the parameters is shown in FIG. 40B. In this example, each piece of the timbre data for each sampling frequency FS that can be set is prepared for the same tone control parameter VATONEPARi. Namely, for VATONEPAR1(FS1, FS2) through VATONEPARm(FS1, FS2), the parameters having the same timbre for each of the sampling frequencies FS1 and FS2 are prepared all in one tone control parameter VATONEPARi. In this case, the timbre parameter corresponding to the sampling frequency FS is extracted from one tone control parameter VATONEPARi, and the extracted parameter is stored in the buffer VAPARBUF. The tone control parameters having the voice numbers subsequent to VATONEPARm + 1 are the tone control parameters having independent timbres corresponding to one of the sampling frequency FS1 and the sampling frequency FS2. Namely, VATONEPARm+1(FS1, *) corresponds only to the sampling frequency FS1. VATONEPARp(*, FS2) corresponds only to the sampling frequency FS2.

In order to prevent changing of the sampling frequency from affecting uniqueness of the tone in terms of auditory sensation, the parameters to be adjusted according to the changed sampling frequency include the delay parameters of the delay loop section, the filter coefficients, and the nonlinear characteristics of the nonlinear converter of the exciter.

[0182] FIG. 36 is a flowchart of the physical model sound source processing to be performed in step SS110 of the above-mentioned waveform generation processing. When the physical model sound source processing is started, the delay length setting processing of each variable delay section is performed in step SS120 according to the designated pitch frequency, the operation sampling frequency SAMPFREQ indicating the setting state of each section, and the tone control parameter VATONEPAR stored in the buffer VAPARBUF. Each delay time length is set as shown in FIG. 38. Then, in step SS121, the computation processing associated with the exciter as shown in FIG. 26 is performed based on the operation sampling frequency SAMPFREQ, the signal P of breath pressure or bow velocity, the signal E of embouchure or bow pressure, and the tone control parameter VATONEPAR stored in the buffer VAPARBUF. Namely, the exciter return signal EX IN is captured. Then, based on the filter parameter FLTPAR corresponding to the operation sampling frequency SAMPFREQ, filter computation of the exciter filter FIL10 is performed. Further, computation of the nonlinear converter 1 is performed by the nonlinear conversion characteristics corresponding to the operation sampling frequency SAMPFREQ. If required, computation of the nonlinear converter 2 is performed. Also, computation of portions peripheral to these converters is performed. Then, the exciter output signal EX OUT is generated and outputted.

[0183] In step SS122, computation processing associated with the tube/string model shown in FIG. 24 is performed based on the operation sampling frequency SAMPFREQ and the parameter VATONEPAR stored in the buffer VAPARBUF. Namely, the exciter output signal EX OUT is captured, and computation of the junction section is performed based on the junction parameter JUNCTPAR corresponding to the operation sampling frequency SAMPFREQ. Further, computation of the delay loop section is performed. Based on the filter parameter FLTPAR corresponding to the operation sampling frequency SAMPFREQ, computations of the terminal filters FILTER-R and FILTER-L are also performed. Then, the generated exciter return signal EX IN and the output sample signal OUT are outputted.

[0184] In step SS123, computation of the timbre effector as shown in FIG. 25 is performed based on the operation sampling frequency SAMPFREQ and the parameter VATONEPAR stored in the buffer VAPARBUF. Namely, the output sample signal OUT is taken out, and computations of the envelope controller EL, the resonator model section RE, and the effector EF are performed, respectively. Then, the generated final output is outputted as the tone waveform data TONEOUT. This tone waveform data TONEOUT is written into the waveform output buffer WAVEBUF in response to the sampling frequency FS as described above.

[0185] FIG. 37 is a flowchart of the delay loop computation processing performed in step SS122 of the physical model section computation processing. This flowchart shows in detail only the computation processing associated with the terminal filter FILTER-R and the multiplier MU8. The computation processing of the FILTER-L and the multiplier MU3 is performed in the same manner. When the delay loop computation processing is started, computation of the loop up to the right-side end immediately before the terminal filter FILTER-R is performed in step SS130. Then, the computation skip condition is checked in step SS131. This check is performed to skip the computation of the section of which loop gain is substantially zero, thereby saving the total computation amount. Specifically, there are three computation skip conditions. The first computation skip condition is that the output of the terminal filter FILTER-R is 0. This condition may also be that the value 0 is continuously outputted from the terminal filter FILTER-R for a predetermined time. Further, the input of the terminal filter FILTER-R and the contents of the internal delay register may be checked. This condition may also be satisfied when the final output TONEOUT is sufficiently attenuated. The second computation skip condition is that the input signal of the terminal filter FILTER-R is not substantially changed. In this case, the computation is skipped and the output value from the immediately preceding terminal filter FILTER-R is assumed to be the current output value. Further, the immediately preceding output value may be the current output value also in the multiplier MU8. The third computation skip condition is that the multiplication coefficient TERMGR of the multiplier MU8 is zero or nearly zero. In this case, the computation is skipped and the right-side output is made zero.

[0186] When any of the above-mentioned computation skip conditions that is associated with the terminal filter FILTER-R has been satisfied, the decision is made YES in step SS132. Then, in step SS133, processing for passing the output value corresponding to the satisfied condition is performed. If the computation skip condition associated with the terminal filter FILTER-R is found not satisfied, the computation associated with the terminal filter FILTER-R is performed in step SS137. When the processing in step SS133 or SS137 has been completed, it is determined in step SS134 whether the computation skip condition associated with the multiplication coefficient TERMGR is satisfied. If this condition is found satisfied, the decision is YES. Then, in step SS135, the processing for passing the output value corresponding to the satisfied condition is performed. If the condition is found not satisfied, computation for multiplying the multiplication coefficient TERMGR in the multi-

plier MU8 is performed in step SS138. When the processing of step SS135 or SS138 has been completed, computation processing of the remaining delay loop portions is performed in step SS136, upon which control is returned.

[0187] Computation may be skipped not only with the delay loop but also with the exciter or the timbre effector. For the exciter, whether the computation is to be skipped or not is determined by checking the signal amplitude of the signal path and the associated parameters if the values of the amplitude and the parameters are nearly zero. For the timbre effector, when the output of the envelope controller EL, the resonator model section RE, or the effector EF has been sufficiently attenuated to nearly zero, the computation for each block of which output is nearly zero may be skipped to make the output value zero. In the second embodiment described so far, control of changing the sampling frequency FS may cause an aliening noise depending on the nonlinear conversion characteristics in the nonlinear section. This problem may be overcome by performing oversampling on the input side of the nonlinear conversion and by band-limiting the obtained nonlinear conversion output by a filter to return the sampling frequency to the original sampling frequency.

[0188] If a new key-on occurs during the current key-on state in the physical model sound source shown in FIG. 24, processing for sounding the music tone corresponding to the new key-on is performed. If the sounding is made by inheriting the music tone corresponding to the preceding key-on, the signals that circulate inside the physical model, for example, the signals inside the delay sections such as the tube/string model section may be basically handled without change. New exciter signals may only be generated according to the new key-on. If a highly independent music tone is set up without making such inheritance, or a music tone having a timbre different from that of the immediately preceding key-on is to be sounded in response to the new key-on, the delay circuit in the physical model sound source must be initialized or reset according to the new key-on. In this case, if the number of sound channels in the physical model sound source is one, the delay area in the RAM 3 constituting all delay circuits on the physical model sound source are cleared and initialized to generate the music tone corresponding to the new key-on. If the number of sound channels in the physical model sound source is plural, the delay area in the RAM 3 constituting the delay circuit for the sound channel attenuated most is cleared to mute the music tone of that sound channel. Then, using the initialized delay area, the music tone corresponding to the new key-on is generated.

[0189] Clearing the delay area in the RAM 3 is realized by writing data "0" to that area, so that the music tone generation is unnaturally delayed by the time of clearing. FIG. 41 shows a hardware constitution of a delay circuit that can eliminate the wait time for clearing the delay area. As shown in FIG. 41, the delay circuit is made up of two systems of delay means. The delay means of the first delay system is composed of a multiplying means MU31, a delay means DELAYa, and a multiplying means MU32 interconnected in series. The delay means of the second delay system is composed of a multiplying means MU33, a delay means DELAYb, and a multiplying means MU34 interconnected in series. Input data INPUT is inputted in both the first and second delay systems. The outputs of both of the delay systems are added by an adding means AD31, and outputted as delay output data OUTPUT. The multiplying means MU31 is provided with a multiplication coefficient INGAINa, the multiplying means MU33 is provided with a multiplication coefficient INGAINb, the multiplying means MU32 is provided with a multiplication coefficient OUTGAINa, and the multiplying means MU34 is provided with a multiplication coefficient OUTGAINb. As shown in FIG. 41, an input controller is composed of the multiplying means MU31 and MU32. A mixer (MIX) is composed of the multiplying means MU32 and MU34 and the adding means AD31. In FIG. 41, the delay circuit is represented in hardware approach. Actually, the delay circuit is implemented by software, namely a delay processing program that uses the delay area in the RAM 3.

[0190] The following explains the operation of the delay circuit shown in FIG. 41 with reference to FIGS. 42A and 42B. FIG. 42A shows an equivalent circuit for controlling the selection between the first and second delay systems in a selective manner. The input data INPUT is led by a selector (SEL) 31 to the delay means DELAYa or the delay means DELAYb. Namely, the above-mentioned input controller constitutes the selector 31. The capability of the selector 31 is implemented by setting one of the multiplication coefficient INGAINa given to the multiplying means MU31 and the multiplication coefficient INGAINb given to the multiplying means MU33 to "0" and by setting the other multiplication coefficient to "1". The delay output data OUTPUT is outputted from one of the delay means DELAYa and the delay means DELAYb. Namely, the above-mentioned mixer constitutes a selector 32. The capability of the selector 32 is implemented by setting one of the multiplication coefficient OUTGAINa given to the multiplying means MU32 and the multiplication coefficient OUTGAINb given to the multiplying means MU34 to "0" and by setting the other multiplication coefficient to "1". The multiplication coefficient INGAINa and the multiplication coefficient OUTGAINa are controlled to be equal to each other. The multiplication coefficient INGAINb and the multiplication coefficient OUTGAINb are controlled to be equal to each other. Delay amounts DLYa and DLYb according to the pitches of assigned music tones are set to the delay means DELAYa and the delay means DELAYb, respectively.

[0191] The following describes in detail the operation of the delay circuits shown in FIG. 42A. A multiplica-

tion coefficient INPUTa and a multiplication coefficient OUTPUTa are set to "1". A multiplication coefficient INPUTb and a multiplication coefficient OUTPUTb are set to "0". In this case, the input data INPUT is led by the selector 31 to the delay means DELAYa and is delayed by a time corresponding to a delay amount DLYa set by the delay means DELAYa. The delay input data is outputted via the selector 32 as output data OUTPUT delayed by the predetermined time. If the multiplication coefficient INPUTa and the multiplication coefficient OUTPUTa are set to "0" and the multiplication coefficient INPUTb and the multiplication coefficient OUTPUTb are set to "1", the input data INPUT is led by the selector 31 to the delay means DELAYb and is delayed by a time corresponding to a delay amount DLYb set by the delay means DELAYb. The delayed input data is outputted via the selector 32 as output data OUTPUT delayed by the predetermined time.

[0192] The first delay system and the second delay system can be switched to each other n a toggle manner. Therefore, if the first delay system is in use for example when a new key-on occurs, the multiplication coefficient between the multiplication coefficient INPUTa and the multiplication coefficient OUTPUTa in the first delay system is changed from "1" to "0". At the same time, the multiplication coefficient between the multiplication coefficient INPUTb and the multiplication coefficient OUTPUTb in the second delay system is changed from "0" to "1". These changing operations allow the use of the delay means DELAYb in the second delay system. Thus, it is ready to generate the music tone corresponding to the new key-on. Because the multiplication coefficient in the first delay system is changed to "0", data "0" is written to the delay means DELAYa of the first delay system in one period of music tone, thereby clearing this delay means.

[0193] The delay circuit shown in FIG. 42A is represented in hardware approach. When the above-mentioned delay control is performed by software, the selectors 31 and 32 need not be provided on the input side and the output side. The operations equivalent to these selectors can be performed by allocating a free delay area in the RAM 3 every time key-on occurs. When new key-on occurs, the delay means of the delay system to which multiplication coefficient "0" is set shifts by the delay length used so far by the write pointer (or by the memory area allocated to the delay concerned) and is written with data "0" to be cleared. The memory area may be kept in the wait state until the same is allocated with key-on to be generated next. Preferably, a flag is set on this memory area indicating that this area is free. Further, when new key-on occurs, the delay system released by truncate processing may be cleared when the load of the CPU is not heavy.

[0194] The delay circuit shown in FIG. 42B is obtained by replacing the selector 32 of the delay circuit shown in FIG. 42A by a mixer (MIX) 34. The delay circuit of FIG. 42B can perform the same delay control as that of the delay circuit shown in FIG. 42A. In the delay circuit shown in FIG. 42B, the delay systems can be switched by the selector 33 and, at the same time, cross-fade control can be performed in which the multiplication coefficients OUTGAINa and OUTGAINb set, respectively, to the multipliers MU32 and MU34 constituting the mixer 34 are gradually switched from "1" to "0" or from "0" to "1". Within one music tone period, gradual shift can be made from one music tone to another.

[0195] In the delay circuit shown in FIG. 41, the first delay system and the second delay system are always operated in parallel with the multiplication coefficients INGAINa and INGAINb both set to "1" and, every time key-on occurs, a delay amount DLY is set to the delay system other than the delay system assigned to the preceding key-on to provide the pitch corresponding to the new key-on. For example, if the first delay system is assigned to the last key-on, a delay amount DLYb corresponding to the pressing key pitch is set to the delay means DELAYb of the second delay system. At the same time, the multiplication coefficient OUTGAINa of the first delay system is gradually changed from "1" to "0" and the multiplication coefficient OUTGAINb is gradually changed from "0" to "1". When the first delay system and the second delay system are thus cross-fade controlled, the delay amount of the output data OUTPUT outputted from the adding means AD31 substantially changes from the delay amount DLYa to the delay amount DLYb smoothly. Namely, portamento can be adchieved. Further, a music tone of which pitch changes at any pitch curve may be obtained by performing cross-fade control on the first delay system and the second delay system alternately and repeatedly, and by changing arbitrarily, every time cross-fade control is performed, the delay amount DLY set to the delay means of the delay system of which multiplication coefficient gradually changes to "1". Moreover, the first delay system and the second delay system are used as delay circuits corresponding to different sampling frequencies, and the delay amounts of these delay circuits are made equal to each other. Besides, while a sum of the multiplication coefficient INGAINa and the multiplication coefficient INGAINb becomes "1" and a sum of the multiplication coefficient OUTGAINa and the multiplication coefficient OUTGAINb becomes "1", each multiplication coefficient is controlled appropriately. This mixes timbres based on different sampling frequencies, thereby generating a music tone having a new timbre. If the signal amplitude of a branch path in the physical model sound source becomes small, shift from the preceding key-on to the current key-on may shift to the delay system having the lower sampling frequency. When the shift has been completed, the delay system of which assignment has been cleared can be assigned to the other delay circuit.

[0196] The above-mentioned delay circuits are implemented by software by using the delay areas set in the RAM 3. This is schematically illustrated in FIG. 43.

As shown in the figure, a predetermined area in the RAM 3 is assigned to the delay area. This delay area is divided into a plurality of delay areas to provide unit delay areas (DELAY1a, DELAY1b, ..., DELAYA9, ..., DELAYn) for constituting the delay means. These unit delay areas are allocated to the delay means (DELAY1, ..., DELAYn). A flag area may be provided for each of these unit delay areas. A free flag may be set to this flag area, indicating that the unit delay area is not used as a delay means and hence free.

[0197] The following explains the allocation of the delay area for implementing the delay circuit shown in FIG. 41 with reference to FIG. 43. It should be noted that the physical model sound source has first delay circuit through the n-th delay circuit. By the preceding key-on, the unit delay area DELAYa has been allocated to the delay means of the first delay system of the first delay circuit DELAY1 for example, and the delay amount of the unit delay area DELAY1a is set to delay amount DLYi according to the pitch associated with the preceding key-on. Further, by the preceding key-on, the unit delay area DELAY9 has been allocated to the delay means of the first delay system of the n-th delay circuit DELAYn for example, and the delay amount of the unit delay area DELAY9 is set to delay amount DLYi according to the pitch associated with the preceding key-on.

[0198] Next, when the current key-on occurs, the unit delay area DELAY1b is allocated to the delay means of the second delay system of the first delay circuit DELAY1 for example, and the delay amount of the unit delay area DELAY1a is set to delay amount DLYk according to the pitch of the current key-on. By the current key-on, the unit delay area DELAYn is allocated to the delay means DELAYn of the second delay system of the nth delay circuit for example, and the delay amount of the unit delay area DELAYn is set to delay amount DLYk according to the pitch associated with the current key-on. This can perform the operation of the delay circuit shown in FIG. 41.

[0199] The constitution shown in FIG. 43 indicates that the physical model sound source has a single sound channel. FIG. 44 shows the allocation of the delay area for implementing the delay circuit when the physical model sound source has a plurality of sound channels. The following explains the operation of this constitution. When the unit delay area DELAY1a has been allocated to the delay means of the first delay system in the delay circuit DELAY1 of the first channel for example by the preceding key-on, the delay amount of the unit delay area DELAY1a is set to delay amount DLYp according to the pitch of the preceding key-on allocated to the first sound channel. Then, when the current key-on occurs and the unit delay area DELAY1b is allocated to the delay means of the second delay system in the delay circuit DELAY1 of the first sound channel for example, the delay amount of the unit delay area DELAY1a is set to delay amount DLYq according to the pitch associated with the current key-on allocated to the first sound channel. If the unit delay area DELAY9 has been allocated to the delay means of the first delay system in the delay circuit DELAYn of the second sound channel for example by the preceding key-on, the delay amount of the unit delay area DELAY9 is set to the delay amount DLYp according to the pitch associated with the preceding key-on. Then, if the unit delay area DELAYn is allocated to the delay means DELAYn of the second delay system of the second sound channel for example by the current key-on, the delay amount of the unit delay area DELAYn is set to the delay amount DLYq according to the pitch associated with the current key-on. This arrangement allows execution of the operation of the delay circuit shown in FIG. 41 if the physical model sound source has a plurality of sound channels. In the constitutions of FIGS. 43 and 44, the unit delay area to be allocated to each delay circuit may be previously determined in a fixed manner. Alternatively, the allocation may be performed dynamically by checking, every time key-on occurs, the free flag set to the unit delay area.

[0200] As described above, the inventive tone generating method uses a hardware processor having a software module used to compute samples of a waveform for generating a musical tone. The inventive method comprises the steps of periodically providing a trigger signal at a relatively slow rate to define a frame period between successive trigger signals, periodically providing a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period, operating the hardware processor resettable in response to a trigger signal and operable in response to each sampling signal to periodically execute the software module for successively computing a number of samples of the waveform within one frame, and converting each of the samples into a corresponding analog signal in response to each sampling signal to thereby generate the musical tones. The step of operating includes delaying step using a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information. The delay step successively writes the samples of the waveform of one mucical tone into addresses of one of the memory regions, and successively reads the samples from adresses of the same memory region to thereby create the delay. The delay step responds when the hardware processor is reset so that said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from adresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

[0201] Described so far is the software sound source that practices the second preferred embodiment of the invention on a personal computer. In the computer system, this sound source software can be handled

as either application software or device drive software, for example. The way by which the sound source software is to be handled may be appropriately determined according to the system configuration or the operation system OS used.

[0202] The sound source software or the capabilities thereof may be incorporated in another software program such as amusement software, karaoke software, or automatic play and accompaniment software. Also this software may be directly incorporated in the operation system OS. The software according to the present invention can be supplied in a machine-readable disk media such as a floppy disk, a magneto-optical disk, and a CD-ROM or a memory card. Further, the software may be added by means of a semiconductor memory chip (typically ROM) which is inserted in a computer unit. Alternatively, the sound source software associated with the present invention may be distributed through the network I/F 11.

[0203] The above description has been made by using the application on a personal computer for example. Application to amusement equipment such as game and karaoke, electronic equipment, and general-purpose electrical equipment is also practical. In addition, application to a sound source board and a sound source unit is practical. Moreover, application to a sound source machine based on software processing using dedicated MPU (DS) is practical. In this case, if the processing capacity of the MPU is high, the sampling frequency can be raised, thereby multiplying the sampling frequency by n when high-precision waveform output is required. Further, when a plurality of sound channels are used on the sound source, variable control on the sampling frequency and skip control on the computation portion that can be skipped in the computation algorithm may be performed according to the number of channels being sounded. In this case, different sampling frequencies may be set to different performance parts or MIDI channels. Still further, in the above-mentioned embodiment, the sampling frequency of the CODEC is fixed. It will be apparent that this sampling frequency is variable. The sampling frequency is made variable by inserting the processing circuit for matching the sampling frequencies between the waveform output buffer WAVEBUF and the CODEC (DAC) by typically oversampling, downsampling, or data interpolation.

[0204] The present invention is applicable to a software sound source in which the CPU operates in synchronization with the sampling frequency to periodically execute the software module for successively computing waveform samples. For example, the CPU conducts an interrupt for computing one sample at a period of $1/(n \times f_s)$ where n denotes a number of tones and f_s denotes a sampling frequency. Further, the invention is applicable to a hardware sound source using an LSI chip in order to reduce load of ALU and in order to use resources of LSI chip for other tasks than tone generation.

[0205] As described and according to the present invention, music tone waveform generating blocks indicated by a preset algorithm are assigned to selected sound channels, the assigned music tone waveform generating blocks are combined by the algorithm, and music tone waveform generating computation is performed to generate music tone waveform data. Consequently, the number of music waveform generating blocks for the sound channels may be arbitrarily changed before sounding assignment is made. This novel constitution allows, according to the capacity of a music waveform data generating means, flexible adjustment of the load state of the music waveform data generating means and the quality of the music waveform data to be generated.

[0206] The music tone waveform generating blocks indicated by an algorithm set according to the timbre of the music tone are assigned to the selected sound channels. The assigned music tone waveform generating blocks are combined by the algorithm to perform music tone waveform generating computation so as to generate the music tone waveform data.

[0207] Preferably, in setting timbres by a timbre setting means, if the number of music tone waveform generating blocks is set to a performance part concerned by a means for setting number of blocks, the timbre set to that performance part is changed to a timbre defined by music tone waveform generating blocks within that number of blocks. This novel constitution further enhances the above-mentioned effect.

[0208] Preferably, during the music tone waveform generating computation in the sound channel, the number of music tone waveform generating blocks assigned to that sound channel is changed according to a predetermined condition. Consequently, during sounding, the load state of the music tone waveform data generating means and the quality of the music waveform data to be generated may be changed flexibly according to the capacity of that music tone waveform generating means.

[0209] Further, according to the present invention, in a computer equipment which often executes a plurality of tasks such as word processing and network communication in addition to music performance, occurrence of troubles such as an interrupted music tone can be reduced when the CPU power is allocated to the tasks not associated with music performance during processing of the software sound source. In other words, more tasks can be undertaken during the execution of sound source processing.

[0210] Since the present invention is constituted as described above, when the CPU load is high, the sampling frequency can be lowered, thereby generating tone waveform data that prevents the interruption of a music tone. When the CPU load is low, a higher sampling frequency than the normal sampling frequency can be used, thereby generating high-precision tone waveform data. In this case, the number of sound chan-

nels may be changed instead of changing the sampling frequency.

[0211] If a particular condition is satisfied, corresponding computational operations are skipped, so that efficient computation can be performed, thereby preventing the CPU load from getting extremely high. Consequently, the tone waveform data can be generated that prevents the sounding of a music tone from being interrupted. Further, the efficient computation allows the use of the higher sampling frequency than the conventional sampling frequency, resulting in high-precision tone waveform data.

[0212] While the preferred embodiments of the present invention have been described using specific terms, such description is for illustrative purposes only, and it is to be understood that changes and variations may be made without departing from the spirit or scope of the appended claims.

Claims

1. A sound source apparatus having a software module used to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information (MIDI), the apparatus comprising:

   a processor device (1) that periodically executes the software module for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone;
   a detector device for detecting a load of computation imposed on the processor device (1) during the course of generating the musical tone; and
   a controller device operative according to the detected load for changing the variable sampling frequency to adjust a rate of computation of the samples.

2. The sound source apparatus according to claim 1, wherein the controller device provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number.

3. The sound source apparatus according to claim 2, wherein the processor device (1) includes a delay device (SS120) having a memory (20) for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information (MIDI), the delay device (SS120) generating a write pointer for successively writing the samples into addresses of the memory (20) and a read pointer for successively reading the samples from adresses of the memory (20) to thereby create the delay corresponding to an address gap between the write pointer and the read pointer, the delay device (SS120) being responsive to the fast sampling frequency to increment both of the write pointer and the read pointer by one address for one sample, otherwise the delay device (SS120) being responsive to the slow sampling frequency to increment the write pointer by one address n times for one sample and to increment the read pointer by n addresses for one sample.

4. The sound source apparatus according to claim 2, wherein the processor device (1) includes a delay device (SS120) having a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information (MIDI), the delay device (SS120) successively writing the samples of the waveform of one mucical tone into addresses of one of the memory regions and successively reading the samples from addresses of the same memory region to thereby create the delay, the delay device (SS120) being operative when said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from addresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

5. The sound source apparatus according to claim 1, wherein the processor device (1) executes the software module composed of a plurality sub-modules for successively computing the waveform, the processor device (1) being operative when one of the sub-modules declines to become inactive without substantially affecting other sub-modules during computation of the waveform for skipping execution of said one sub-module.

6. The sound source apparatus according to claim 1, further comprising:

   a provider device for variably providing a trigger signal at a relatively slow rate to define a frame period between successive trigger signals, and for periodically providing a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period; wherein
   the processor device (1) is resettable in response to each trigger signal and operable to periodically execute the software module for successively computing a number of samples of the waveform corresponding to the sampling signals within one frame; and wherein

the controller device is operative according to the detected load for varying the frame period to adjust the number of the samples computed within one frame period, and wherein the apparatus further comprises:

a converter device (14b) responsive to each sampling signal for converting each of the samples into a corresponding analog signal to thereby generate the musical tones.

7. A sound source apparatus having a software module used to compute samples of a waveform in response to a sampling frequency for generating a musical tone according to performance information (MIDI), the apparatus comprising:

processor means (1) to periodically execute the software module for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone;

detector means for detecting a load of computation imposed on the processor means (1) during the course of generating the musical tone; and

controller means operative according to the detected load for changing the variable sampling frequency to adjust a rate of computation of the samples.

8. The sound source apparatus according to claim 7, wherein the controller means provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy such that the rate of the computation of the samples is reduced by 1/n where n denotes an integer number.

9. The sound source apparatus according to claim 8, wherein the processor means (1) includes delay means (SS120) having a memory (20) for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information (MIDI), the delay means (SS120) generating a write pointer for successively writing the samples into addresses of the memory (20) and a read pointer for successively reading the samples from adresses of the memory (20) to thereby create the delay corresponding to an address interval between the write pointer and the read pointer, the delay means (SS120) being responsive to the fast sampling frequency to increment both of the write pointer and the read pointer by every one address for every one sample, otherwise the delay means (SS120) being responsive to the slow sampling frequency to increment the write pointer by every one address at n times for repeatedly writing one sample into consecutive n addresses and to skip the read pointer by consecutive n addresses for reading one sample.

10. The sound source apparatus according to claim 7 further comprising:

provider means for variably providing a trigger signal at a relatively slow rate to define a frame period between successive trigger signals, and for periodically providing a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period; wherein

the processor means (1) is resettable in response to each trigger signal and operable based on each sampling signal to periodically execute the software module for successively computing a number of samples of the waveform within one frame period; and

the controller means is operative according to the detected load for varying the frame period to adjust the number of the samples computed within one frame period, and wherein the apparatus further comprises:

converter means (14b) responsive to each sampling signal for converting each of the samples into a corresponding analog signal to thereby generate the musical tones.

11. The sound source apparatus according to claim 10, wherein the trigger signal is provided periodically by the provider means; and

the processor means (1) includes delay means (SS120) having a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information (MIDI), the delay means (SS120) successively writing the samples of the waveform of one mucical tone into addresses of one of the memory regions and successively reading the samples from adresses of the same memory region to thereby create the delay, the delay means (SS120) being operative when the processor means (1) is reset so that said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from adresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

12. A method using a hardware processor (1) and a software module to compute samples of a wave-

form in response to a sampling frequency for generating a musical tone according to performance information (MIDI), the method comprising the steps of:

periodically operating the hardware processor (1) to execute the software module for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone;
detecting a load of computation imposed on the hardware processor (1) during the course of generating the musical tone; and
changing the variable sampling frequency according to the detected load to adjust a rate of computation of the samples.

13. The method according to claim 12, wherein the step of changing provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy.

14. The method according to claim 12, further comprising the steps of:

variably providing a trigger signal at a relatively slow rate to define a frame period between successive trigger signals;
periodically providing a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period; wherein the hardware processor (1) is resettable in response to each trigger signal and operable based on each sampling signal and is operated to periodically execute the software module for successively computing a number of samples of the waveform within one frame period; and wherein the method further comprises the steps of:
varying the frame period according to the detected load to adjust the number of the samples computed within one frame period, and
converting each of the samples into a corresponding analog signal in response to each sampling signal to thereby generate the musical tones.

15. The method according to claim 14, wherein

the trigger signal variably provided at a relatively slow rate is provided perodically; and wherein the step of operating includes
a delay step (SS120) using a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information (MIDI), the delay step (SS120) successively writing the samples of the waveform of one mucical tone into addresses of one of the memory regions and successively reading the samples from adresses of the same memory region to thereby create the delay, the delay step (SS120) responding when the hardware processor (1) is reset so that said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from adresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

16. A machine readable media for use in a processor machine including a CPU (1), said media containing program instructions executable by said CPU (1) for causing the processor machine having a software module to compute samples of a waveform in response to a sampling frequency for performing operation of generating a musical tone according to performance information (MIDI), wherein the operation comprises the steps of:

periodically operating the processor machine to execute the software module for successively computing samples of the waveform corresponding to a variable sampling frequency so as to generate the musical tone;
detecting a load of computation imposed on the processor machine during the course of generating the musical tone; and
changing the variable sampling frequency according to the detected load to adjust a rate of computation of the samples.

17. The machine readable media according to claim 16, wherein the step of changing provides a fast sampling frequency when the detected load is relatively light, and provides a slow sampling frequency when the detected load is relatively heavy.

18. The machine readable media according to claim 16, wherein the operation further comprises the steps of:

variably providing a trigger signal at a relatively slow rate to define a frame period between successive trigger signals;
periodically providing a sampling signal at a relatively fast rate such that a plurality of sampling signals occur within one frame period; wherein the processor machine is resettable in response to each trigger signal and operable based on each sampling signal and is operated to periodically execute the software module for

successively computing a number of samples of the waveform within one frame period; and wherein the operation further comprises the steps of:

varying the frame period according to the detected load to adjust the number of the samples computed within one frame period, and converting each of the samples into a corresponding analog signal in response to each sampling signal to thereby generate the musical tones.

19. The machine readable media according to claim 18, wherein in the operation the trigger signal provided variably at a relatively slow rate is provided periodically; and the step of operating includes a delaying step using a pair of memory regions for imparting a delay to the waveform to determine a pitch of the musical tone according to the performance information (MIDI), the delay step (SS120) successively writing the samples of the waveform of one mucical tone into addresses of one of the memory regions and successively reading the samples from adresses of the same memory region to thereby create the delay, the delay step (SS120) responding when the processor machine is reset so that said one musical tone is switched to another musical tone for successively writing the samples of the waveform of said another mucical tone into addresses of the other memory region and successively reading the samples from adresses of the same memory region to thereby create the delay while clearing the one memory region to prepare for a further musical tone.

FIG.1

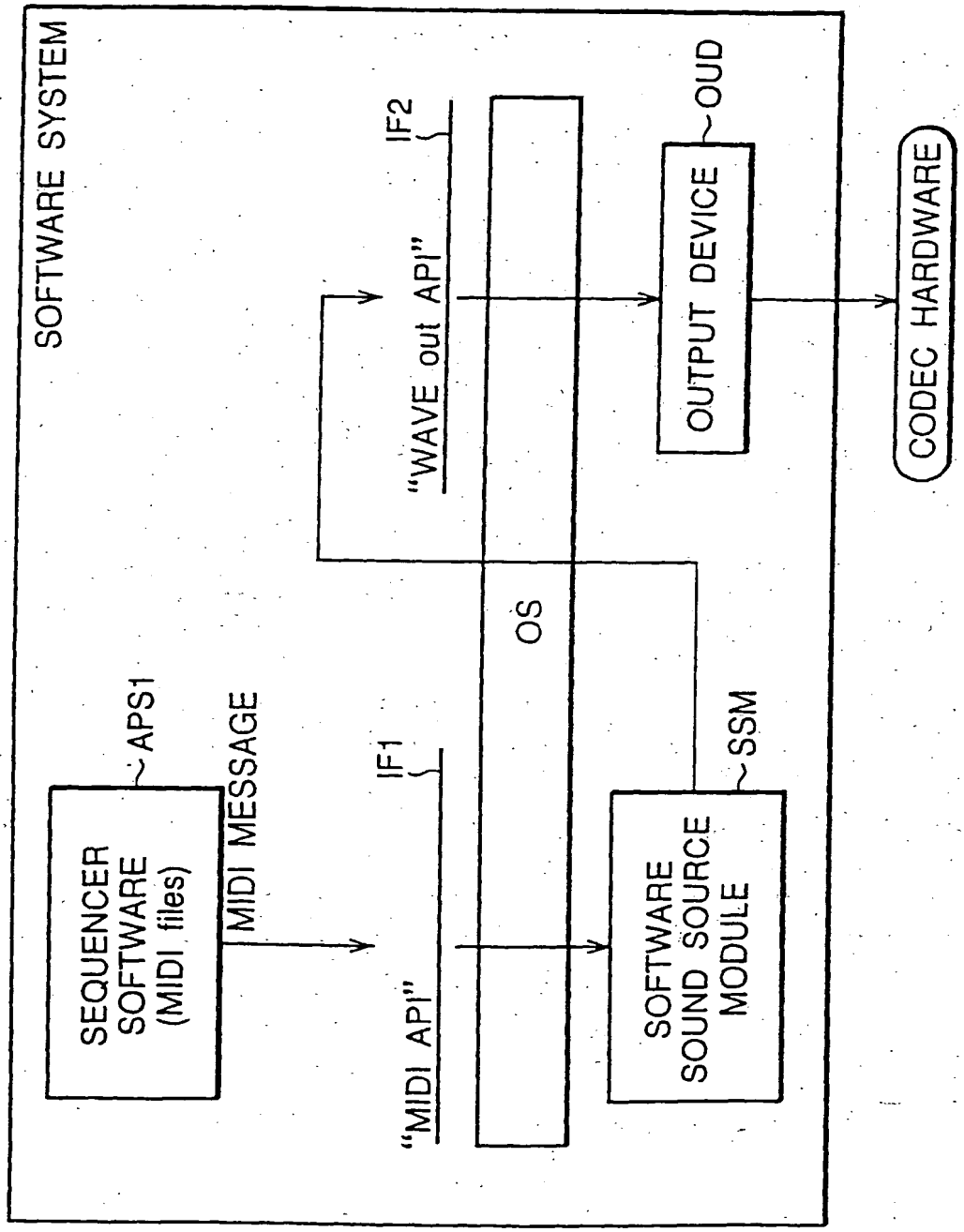


FIG.2

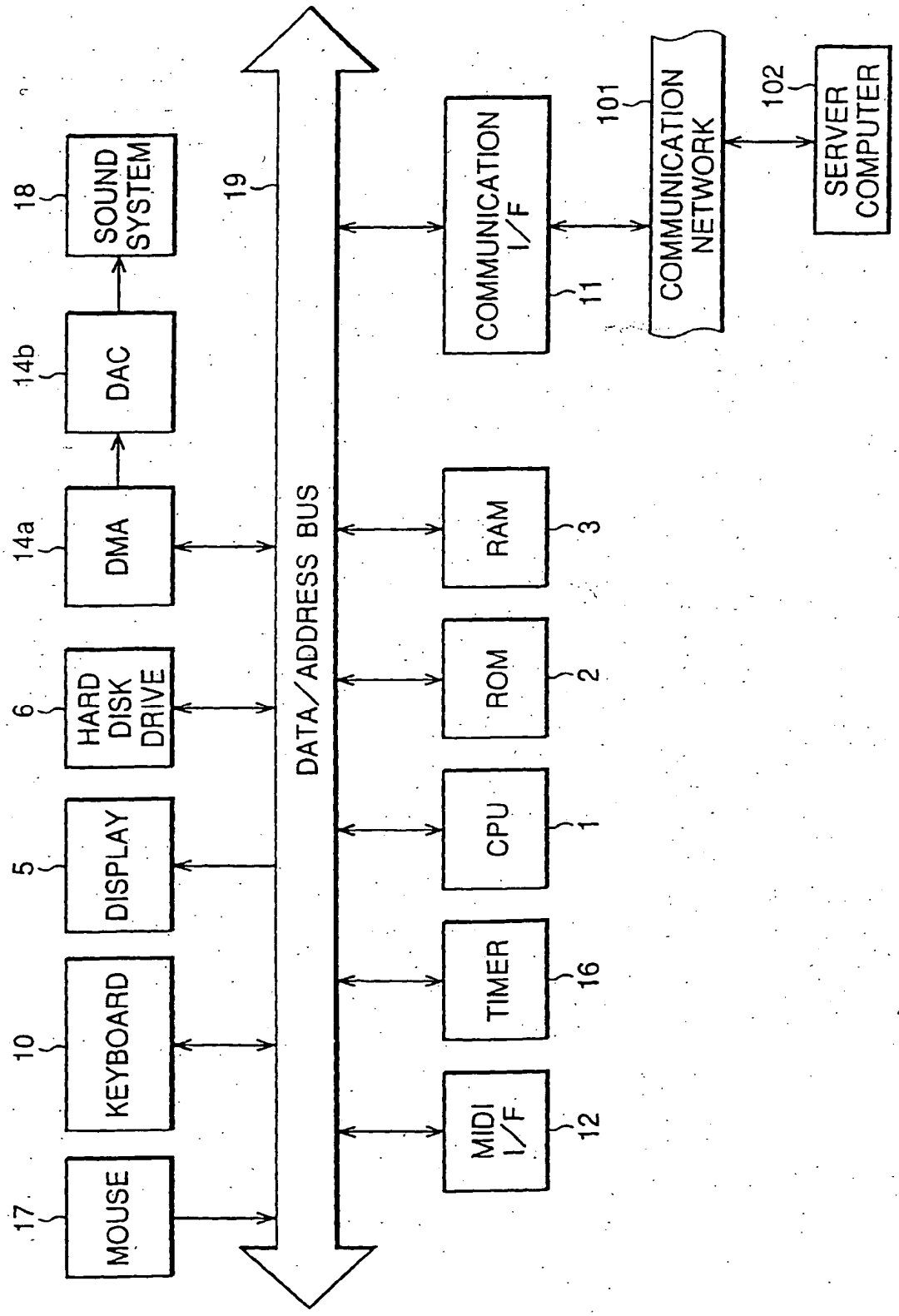


FIG.3

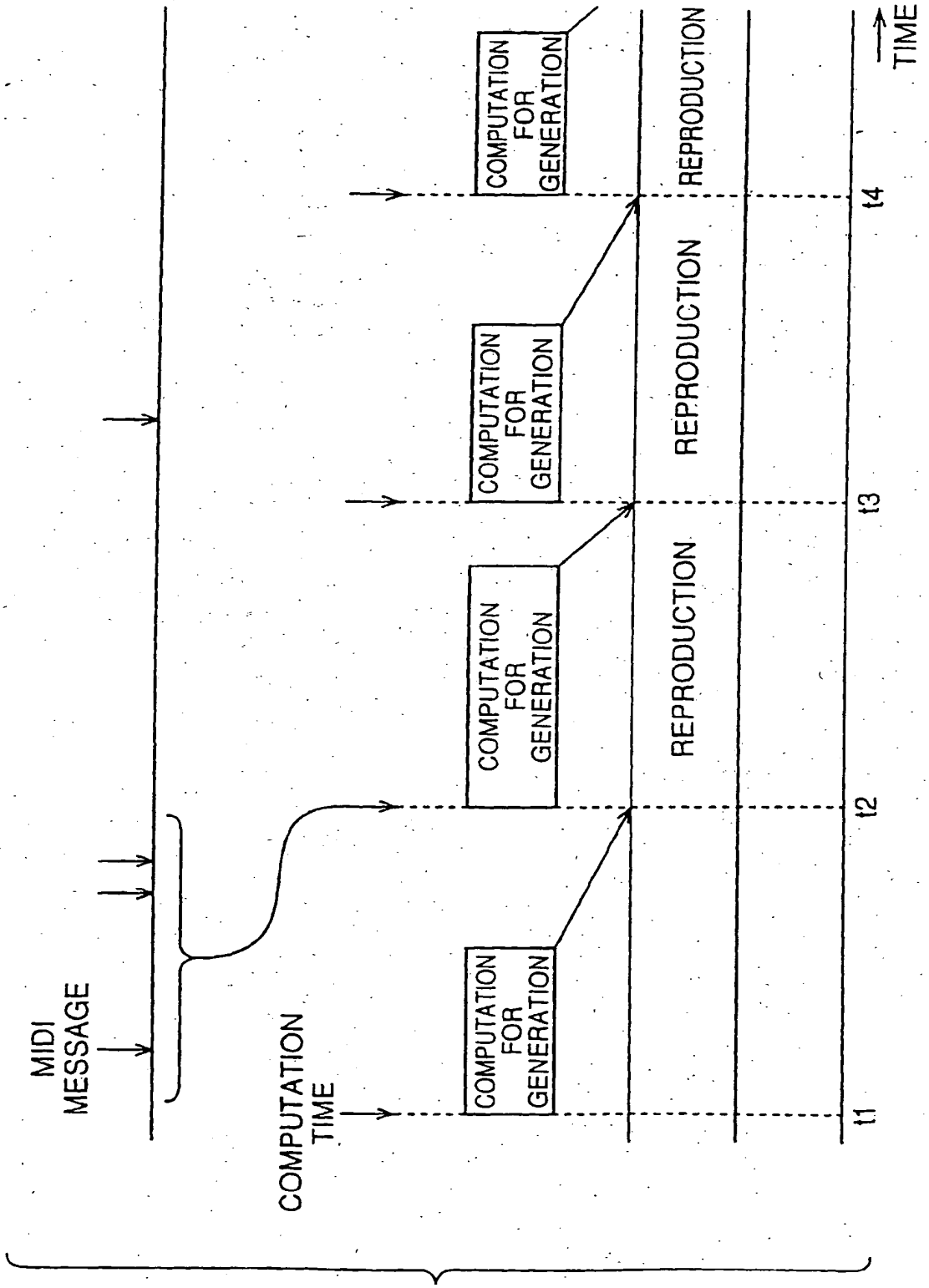


FIG.4A

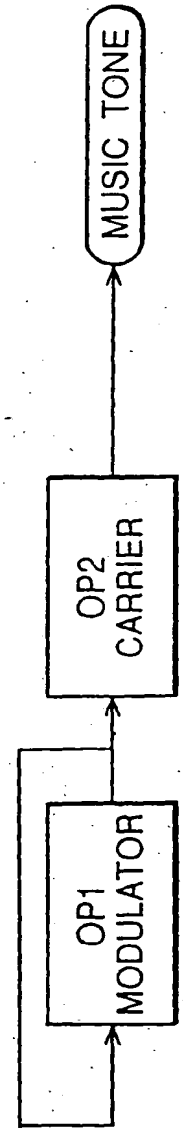


FIG.4B

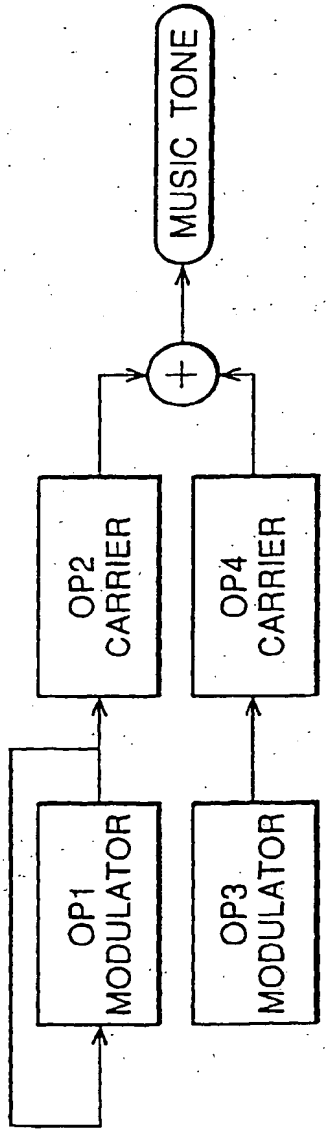


FIG.4C

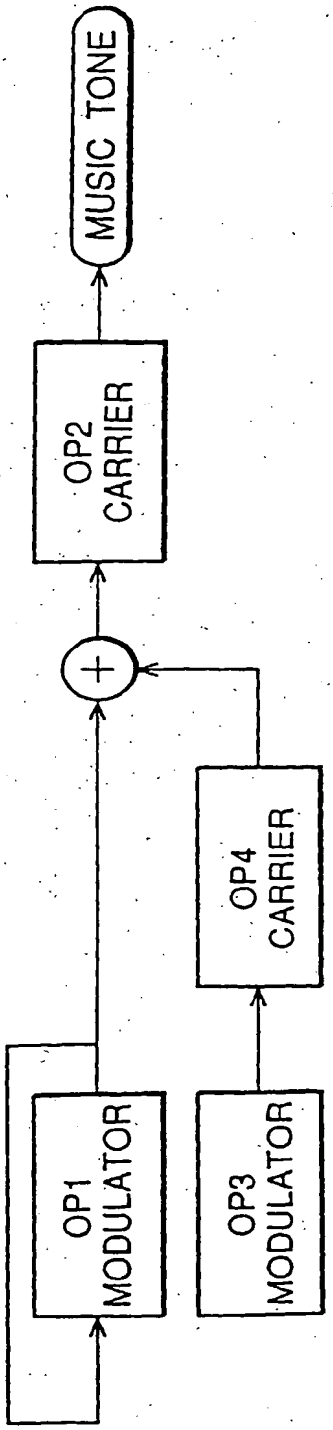


FIG.5

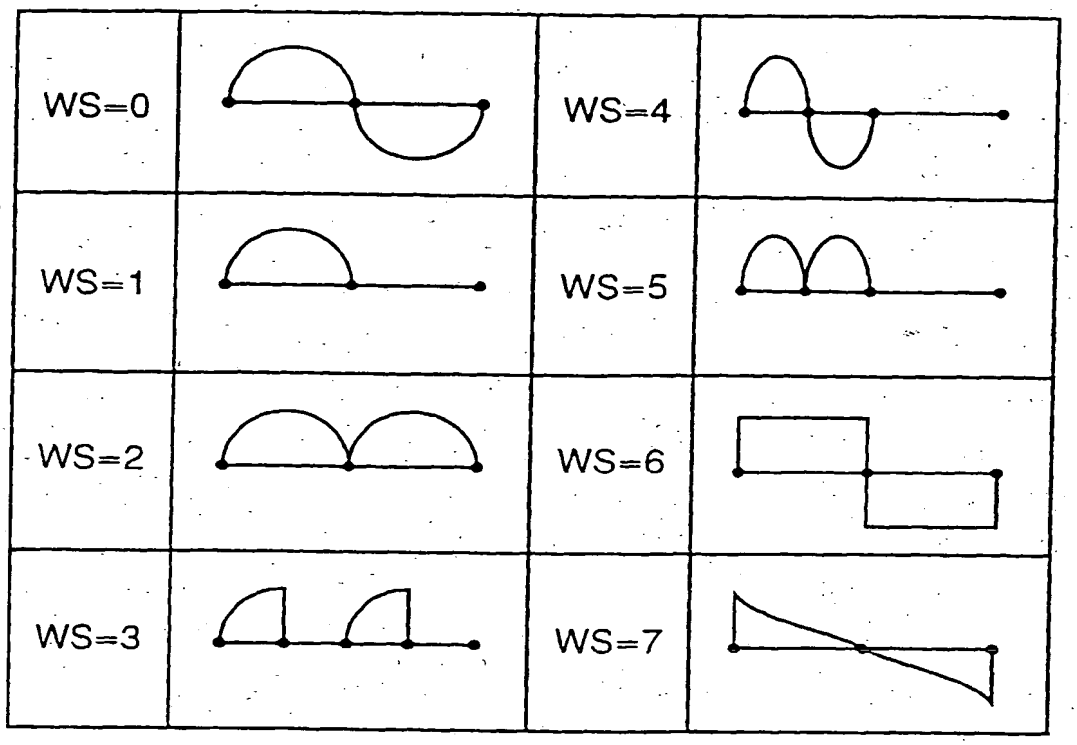


FIG.6

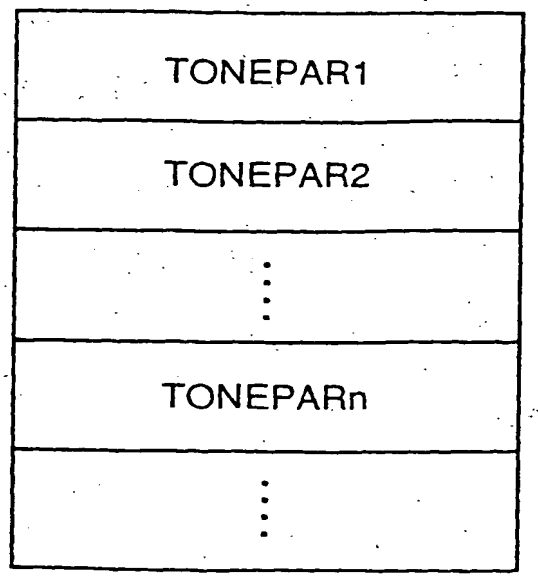


FIG.7A

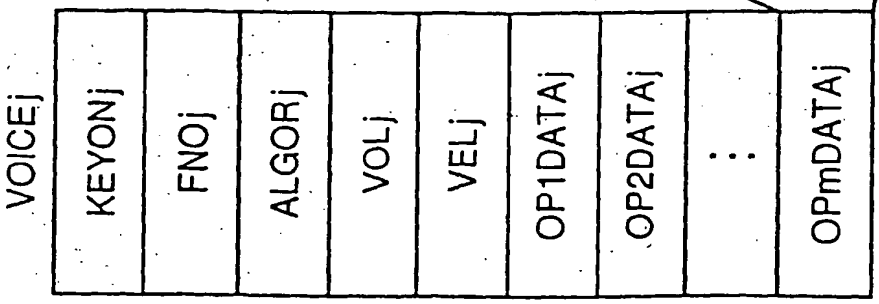


FIG.7B

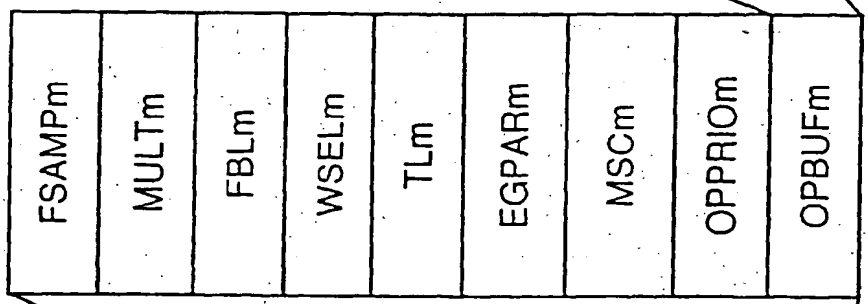


FIG.7C

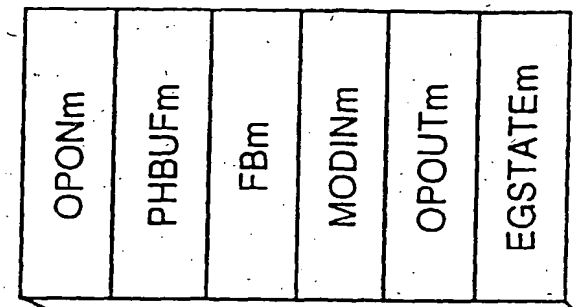


FIG.8

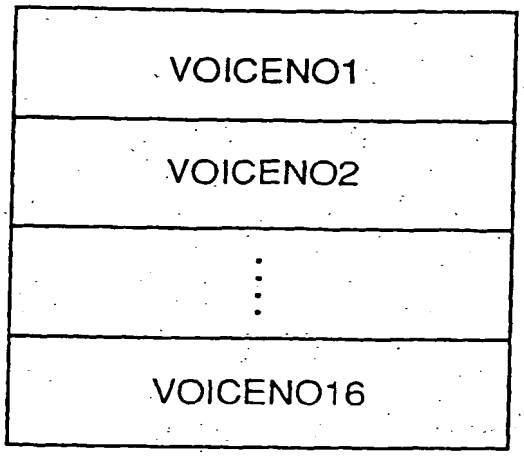


FIG.9

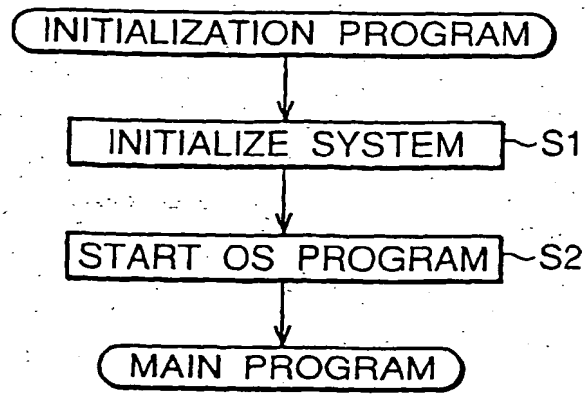


FIG.10

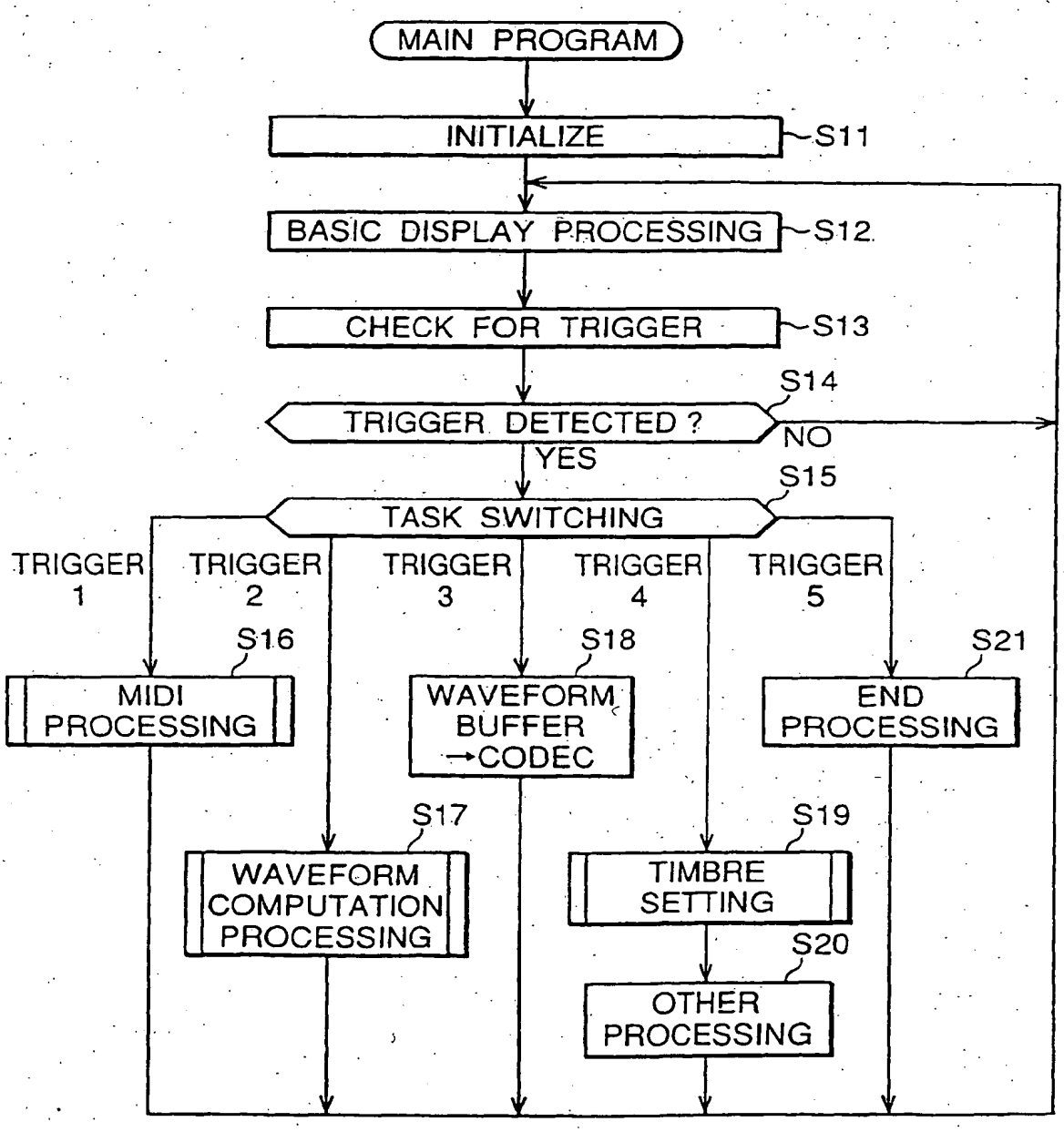


FIG.11

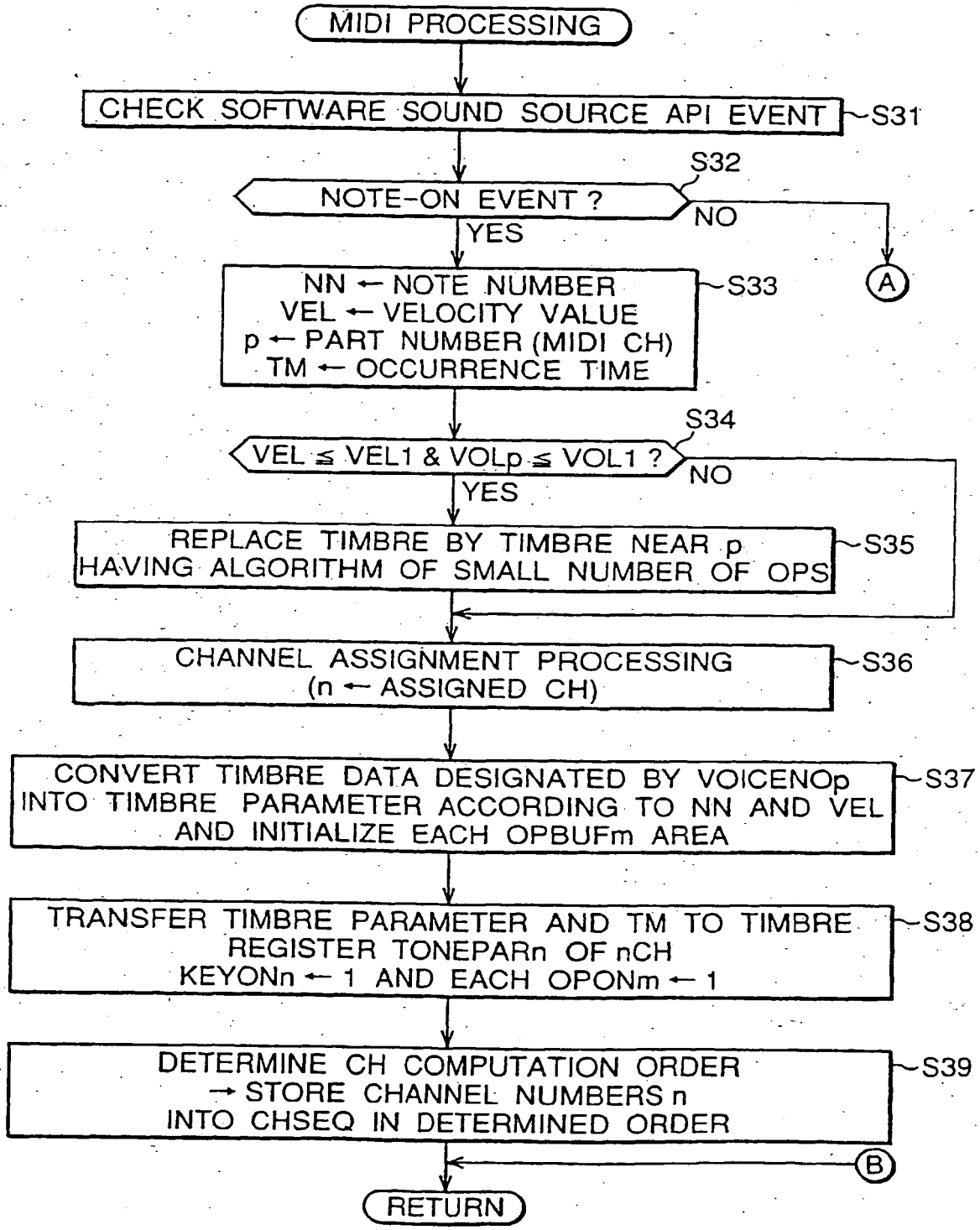


FIG.12

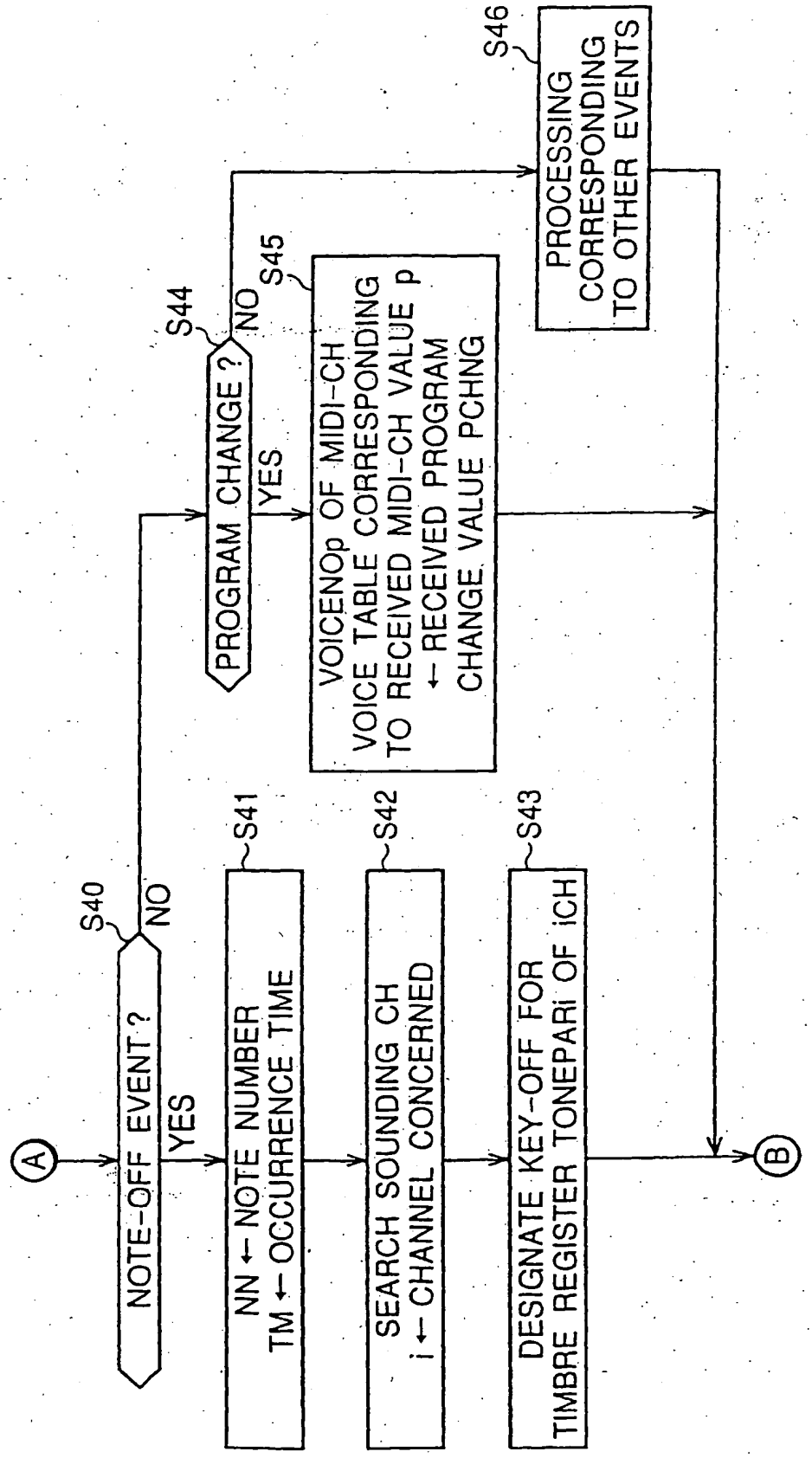


FIG.13

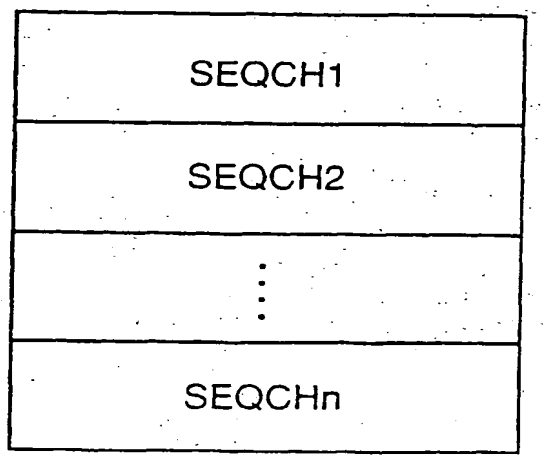


FIG.20

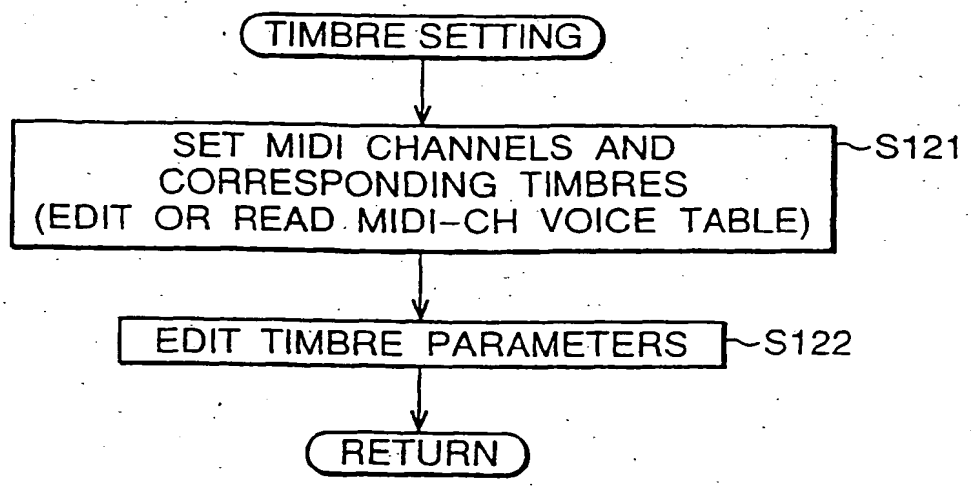


FIG.14

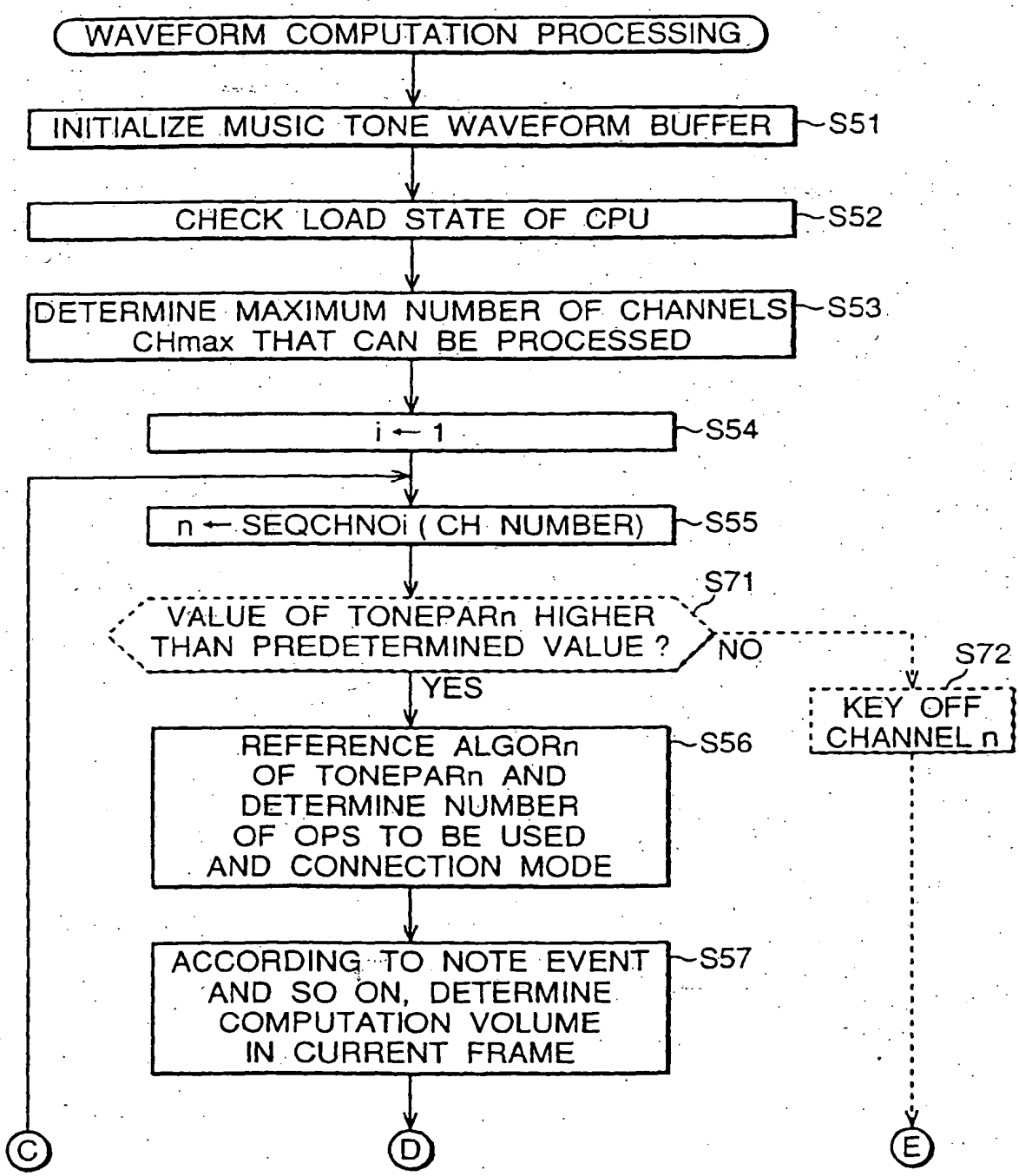


FIG.15

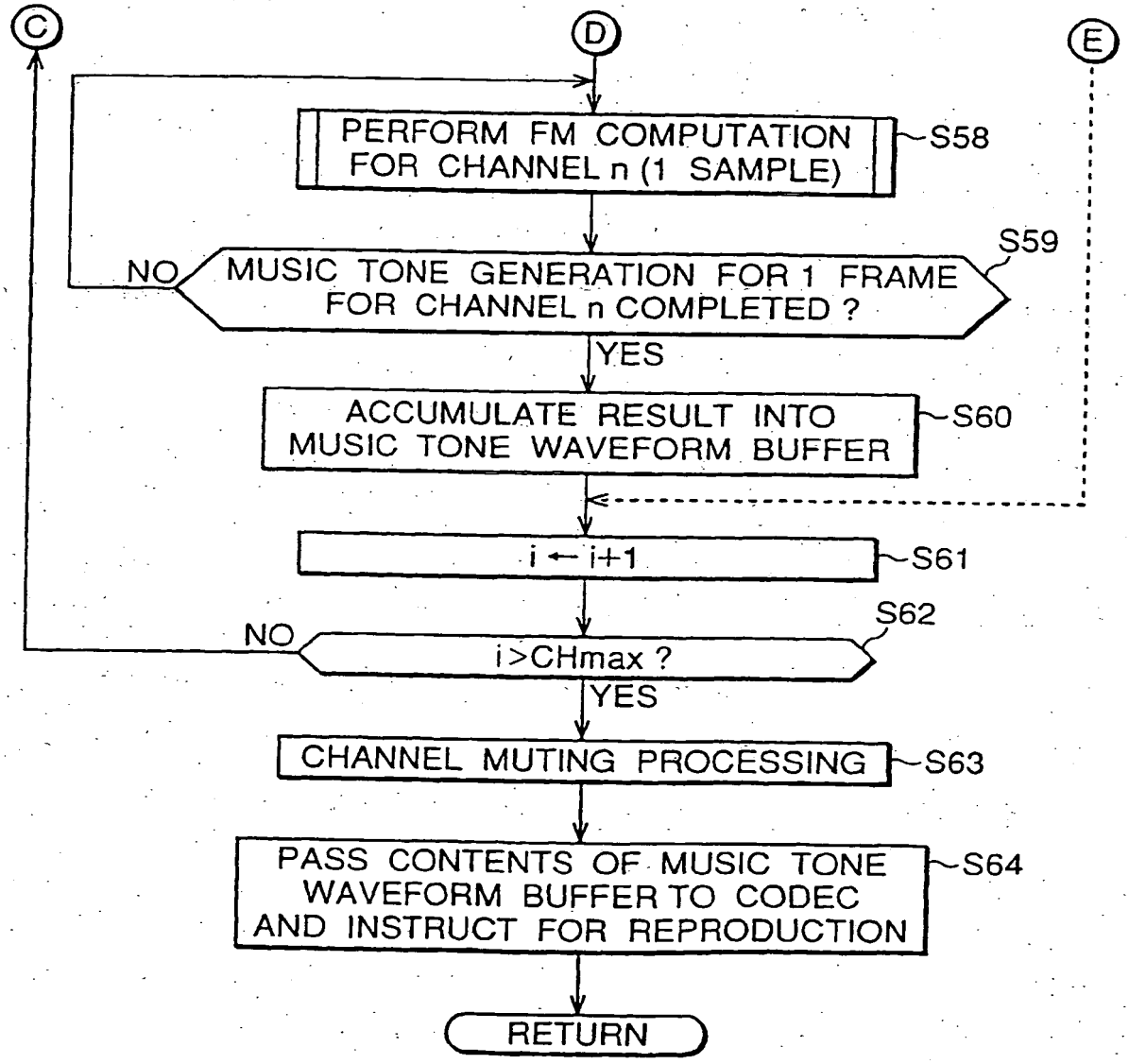


FIG.16

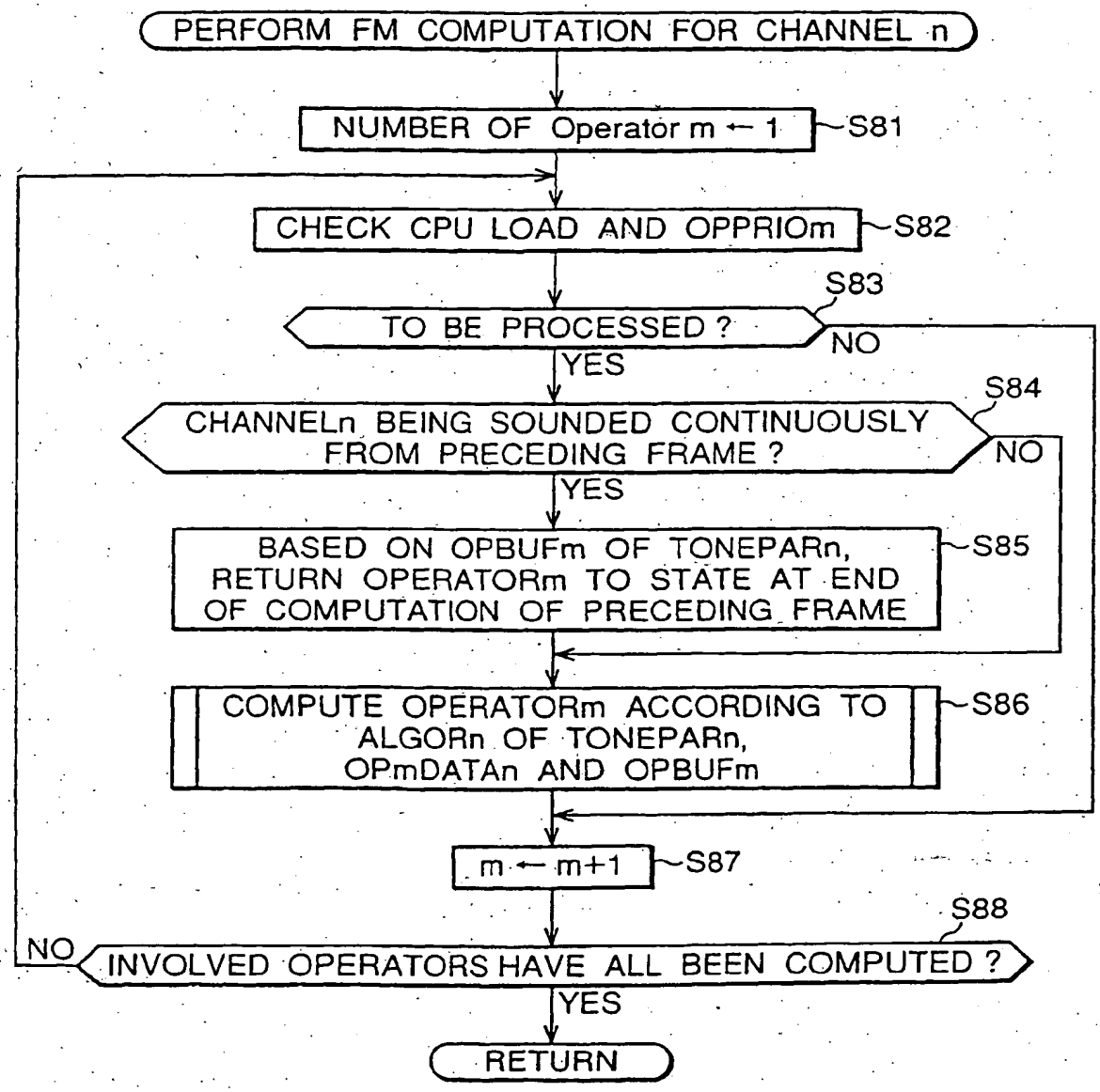


FIG.17

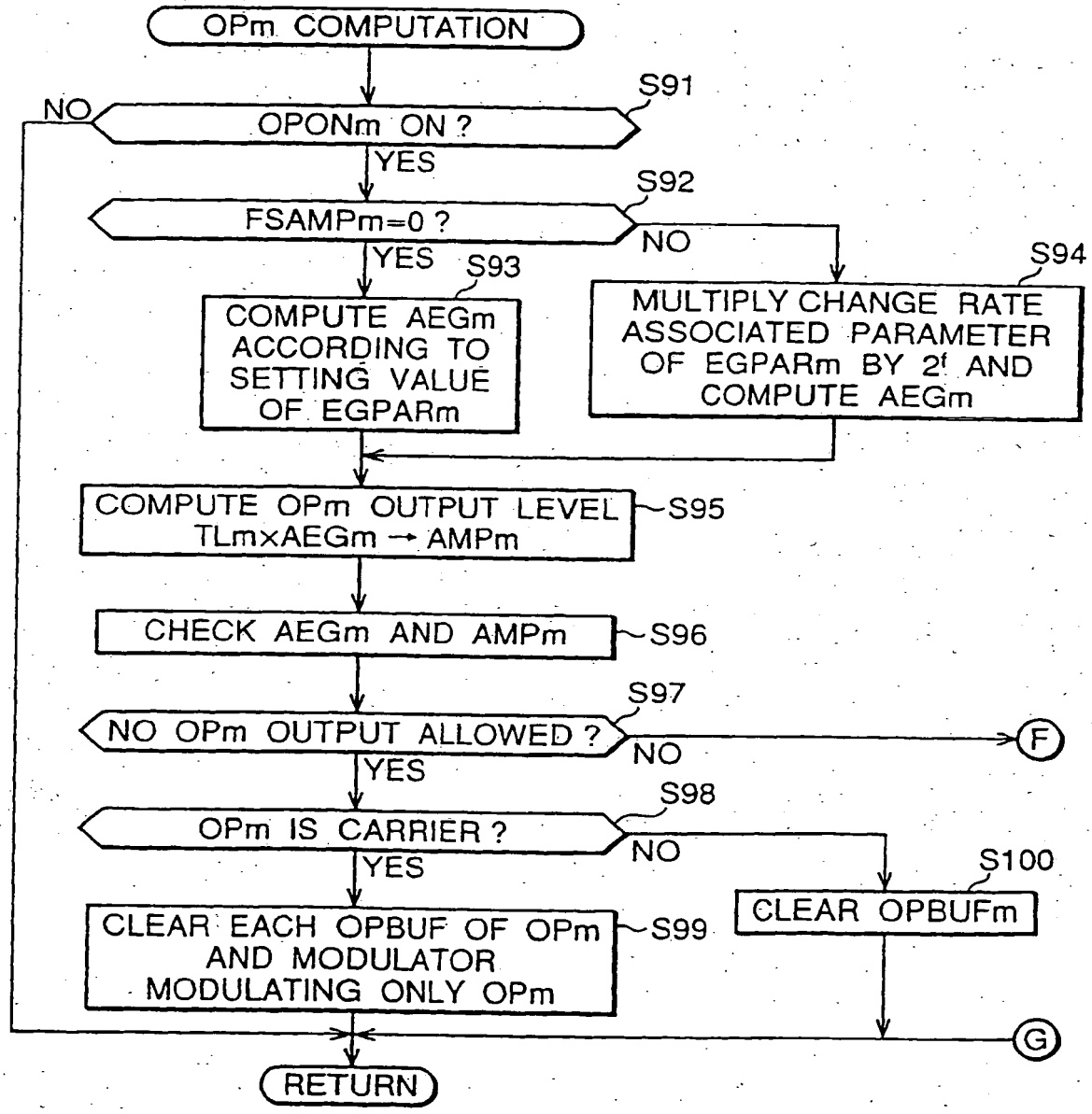


FIG.18

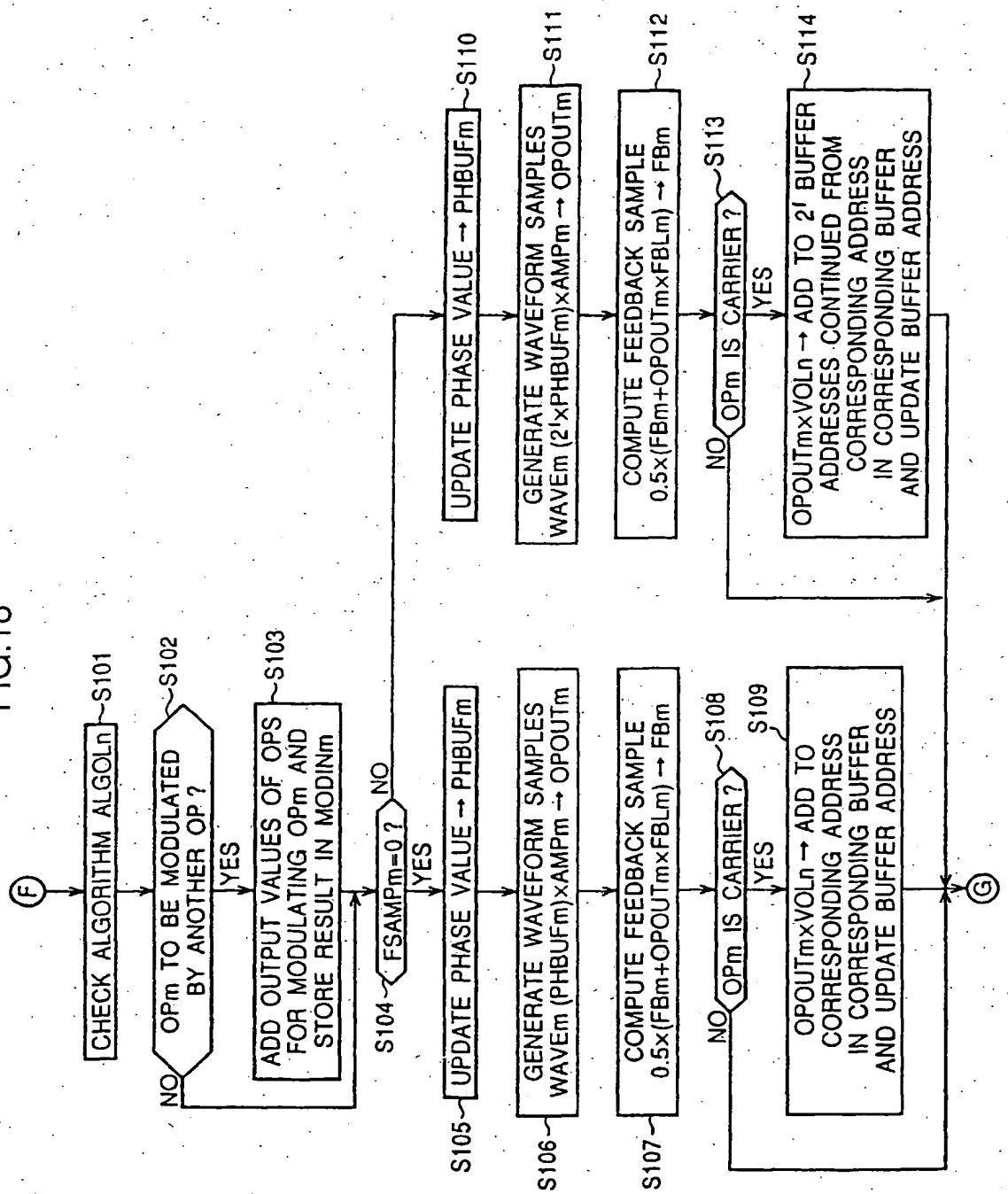


FIG.19

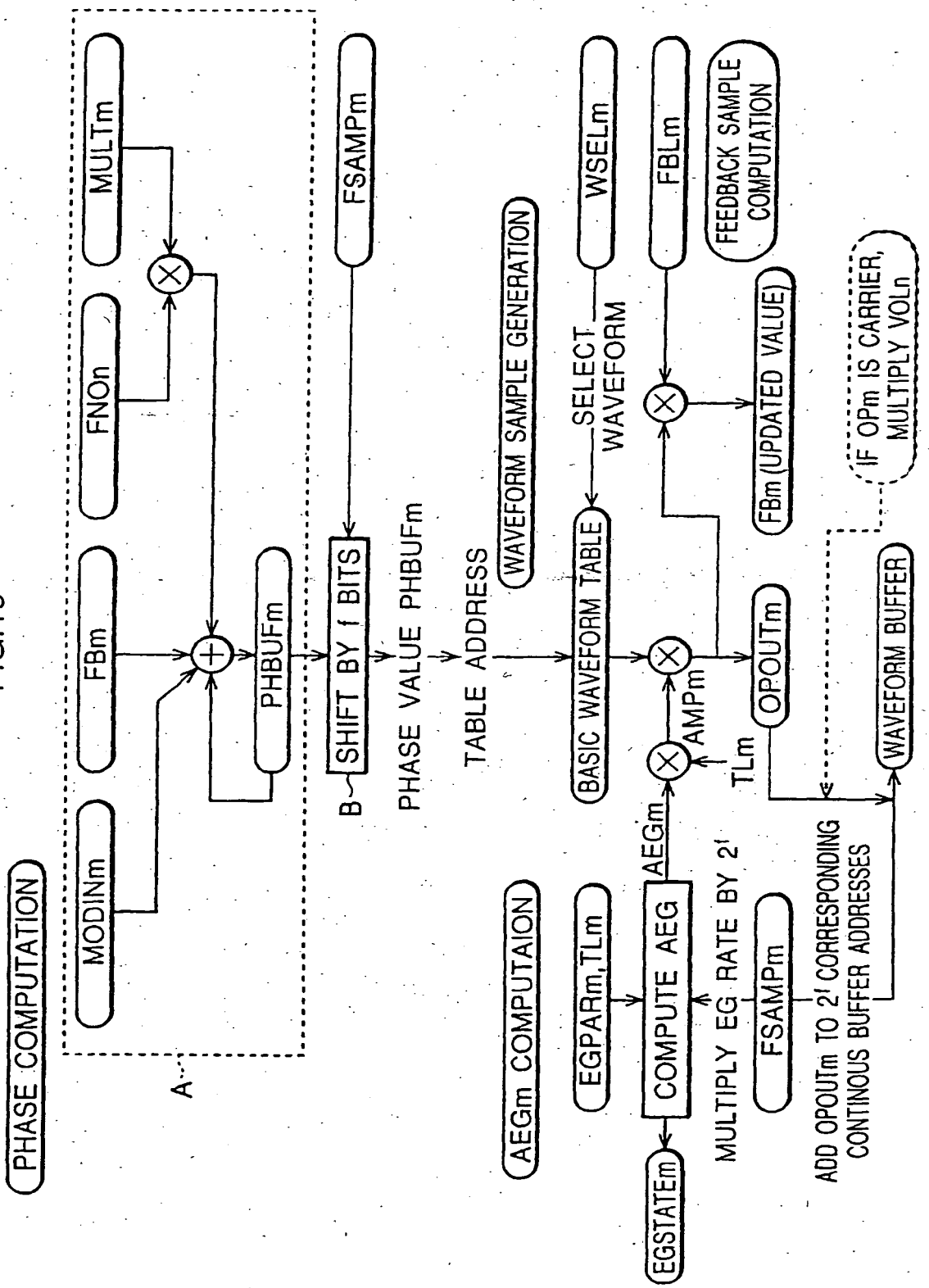


FIG.21

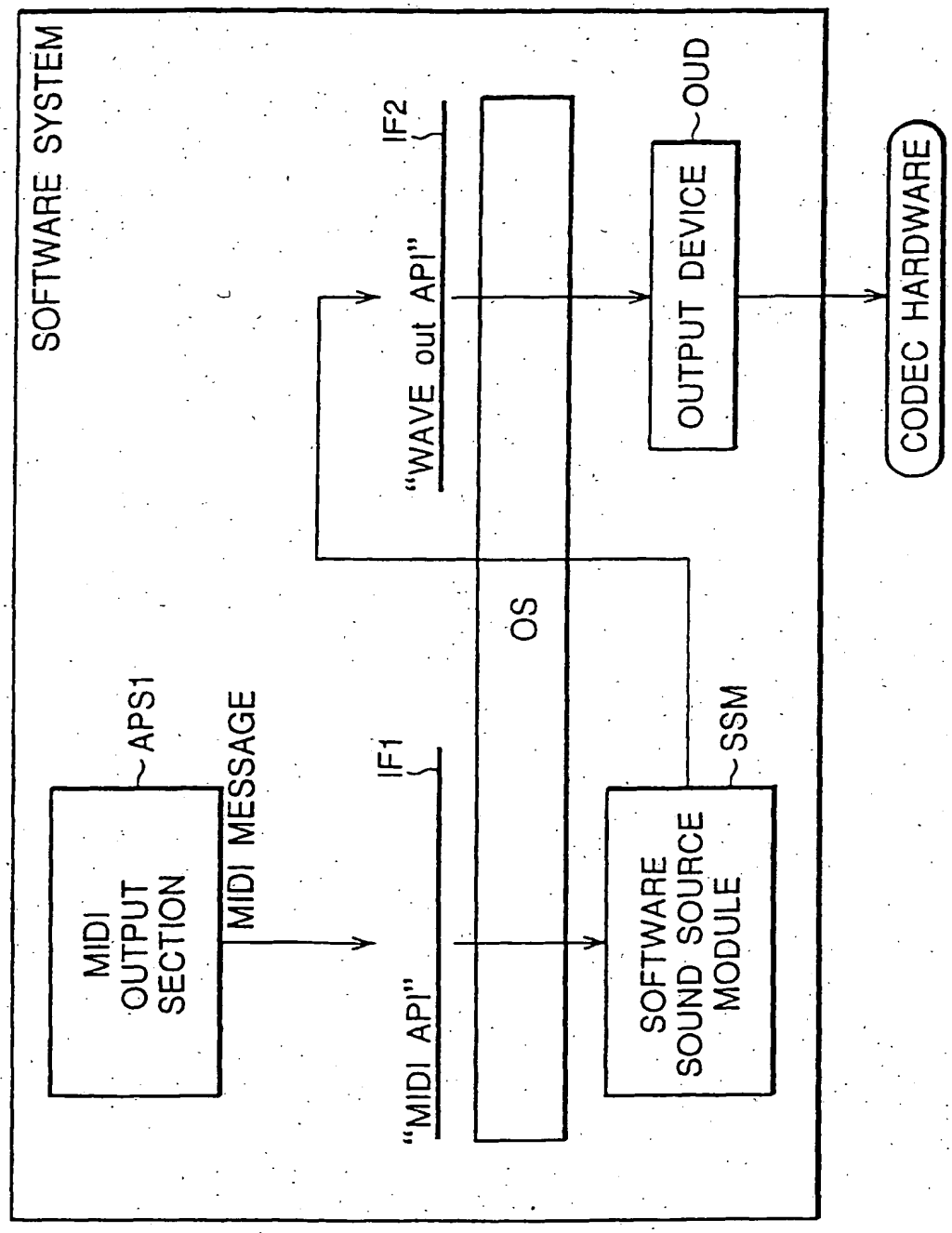


FIG.22

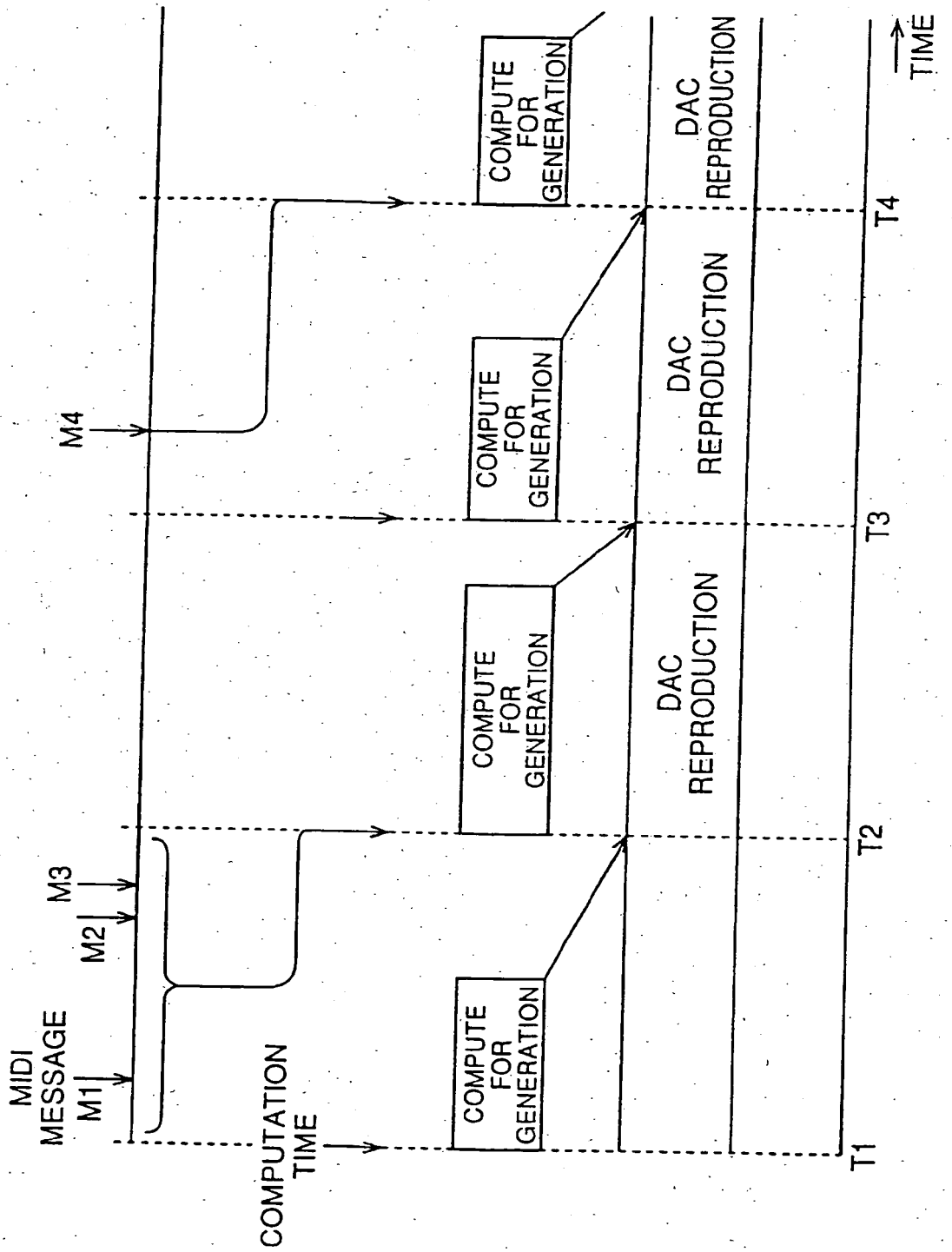


FIG.23

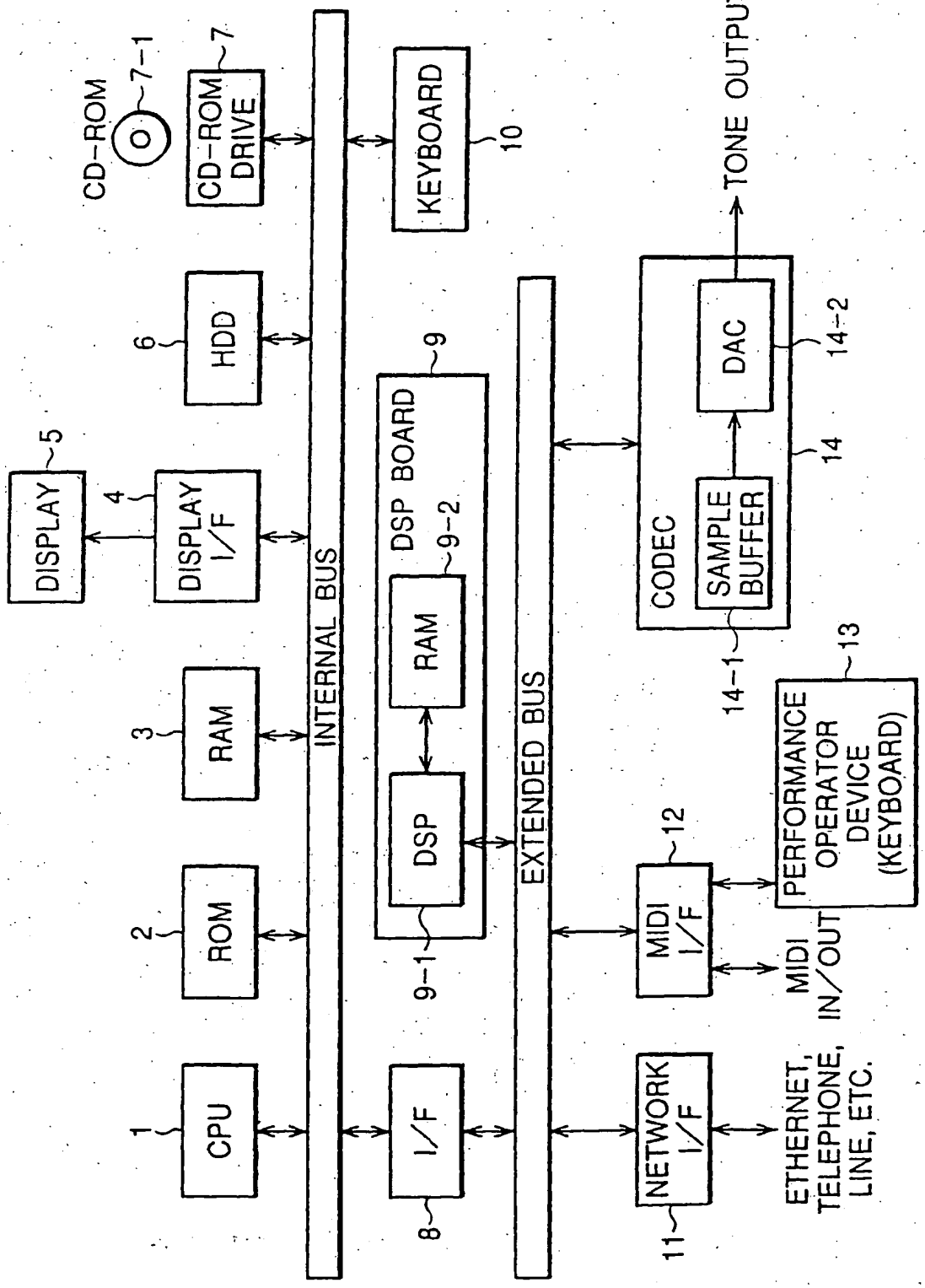


FIG.24

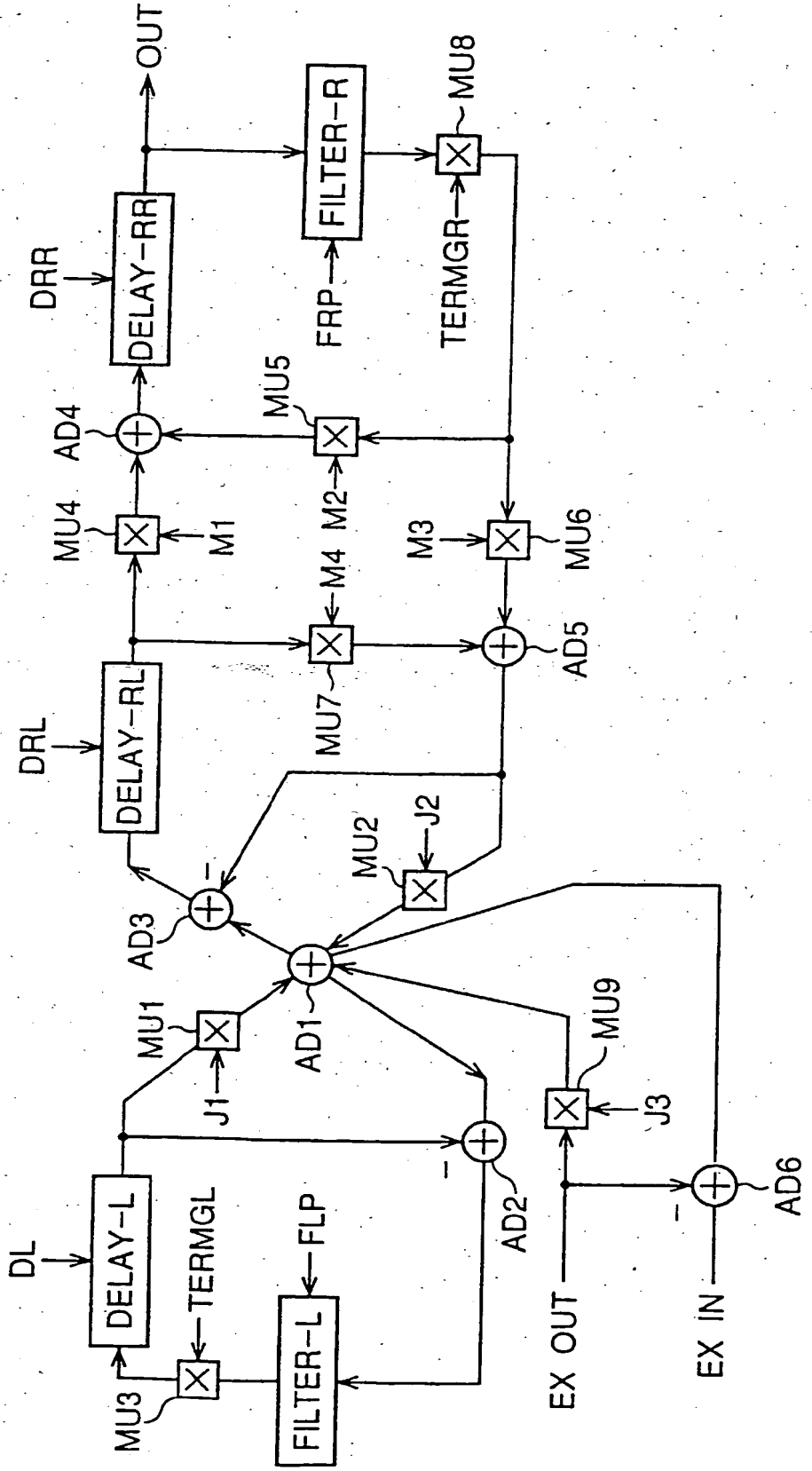


FIG.25

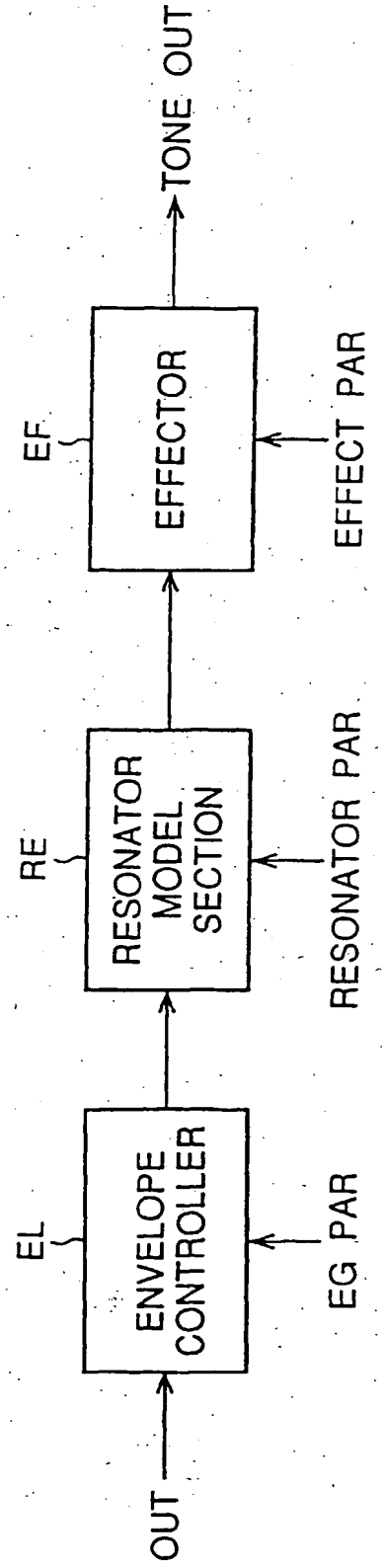


FIG.26

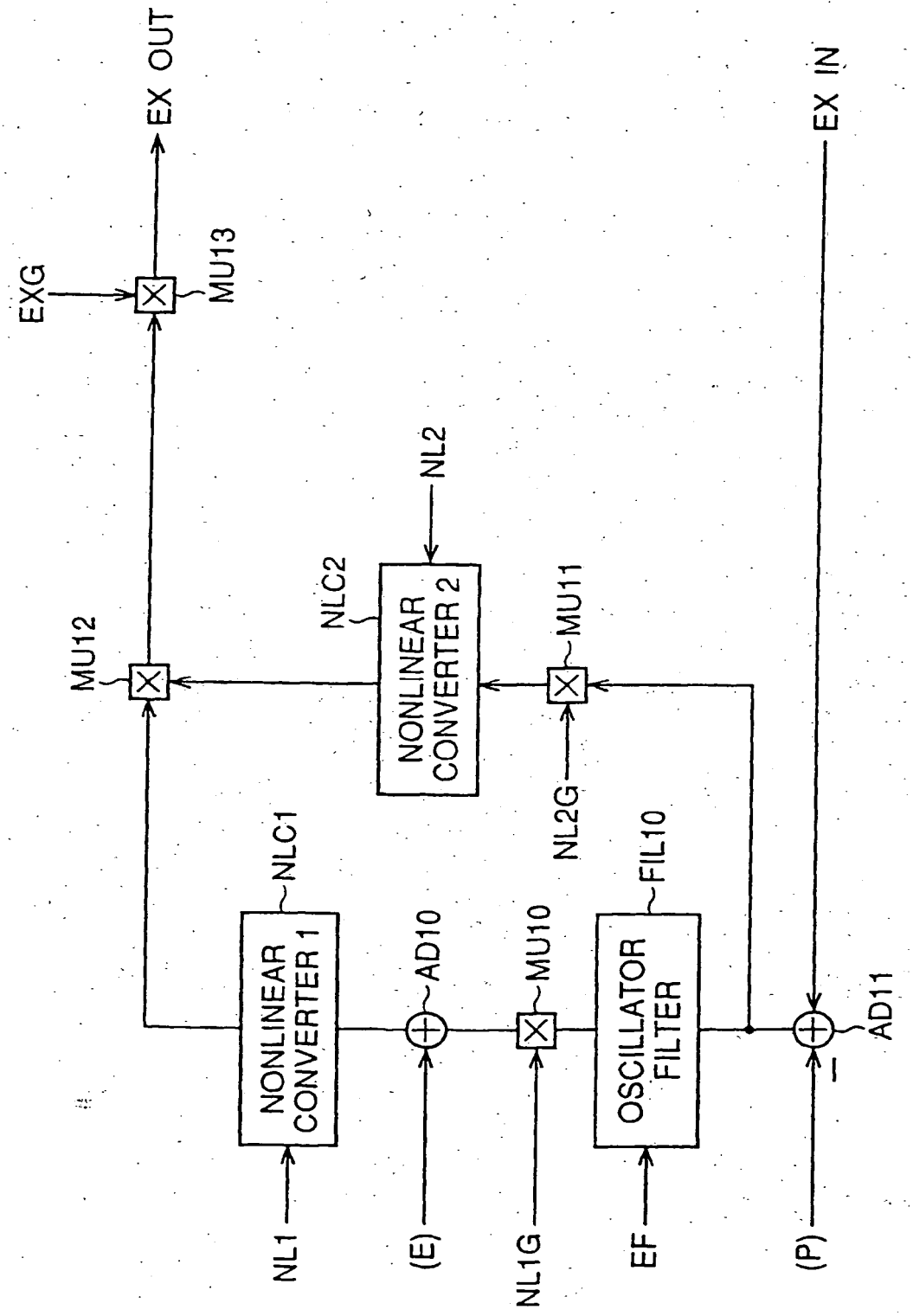


FIG.27

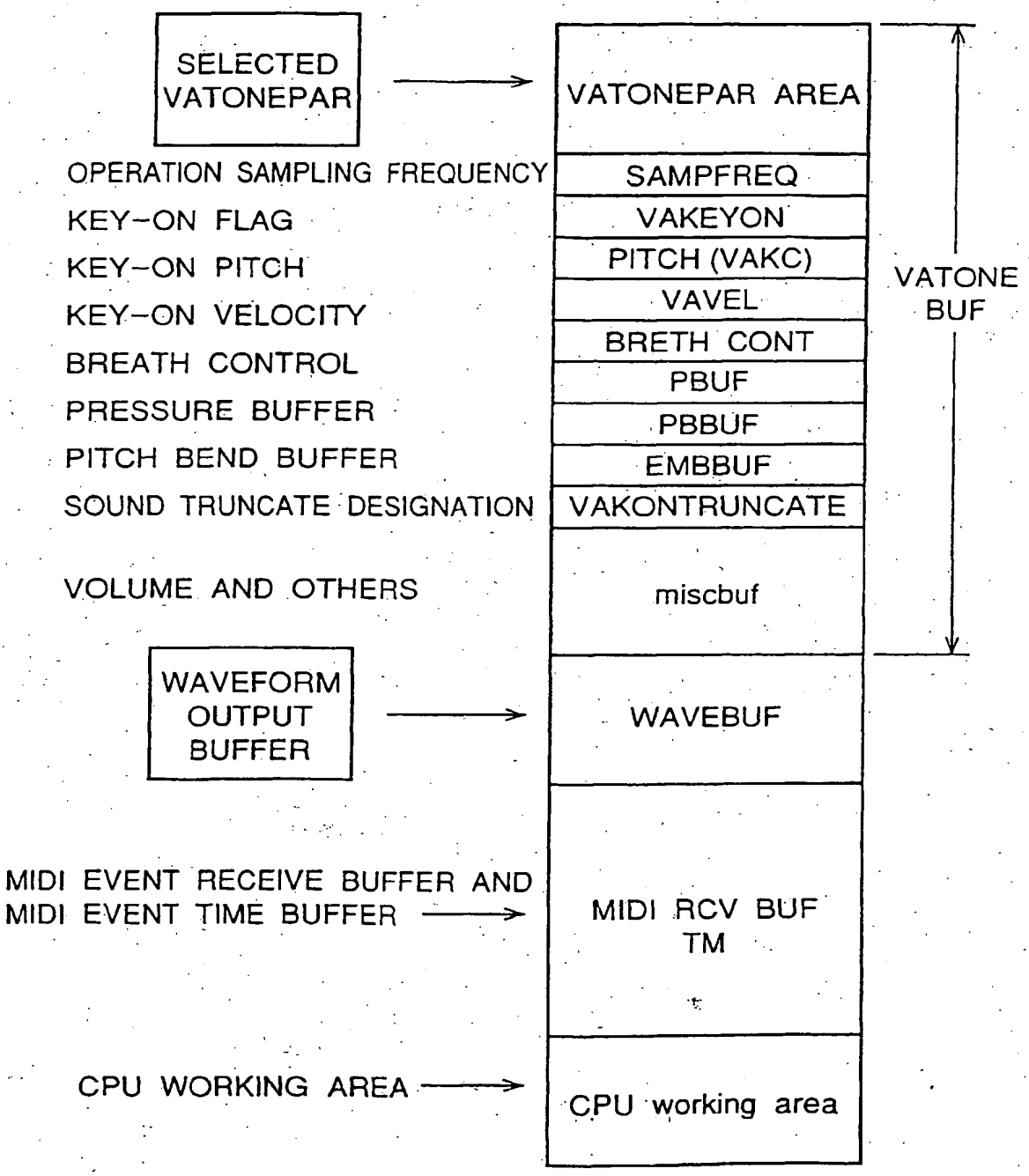


FIG.28

EXICITER PARAMETERS	EF	EXCITER FILTER PARAMETER
	NLG1	NONLINEAR CONVERTER 1 INPUT GAIN
	NLG2	NONLINEAR CONVERTER 2 INPUT GAIN
	EXG	EXCITER OUTPUT GAIN
	NL1	NONLINEAR CONVERTER 1 CHARACTERISTIC PARAMETER (TABLE)
	NL2	NONLINEAR CONVERTER 2 CHARACTERISTIC PARAMETER (TABLE)
P/S PARAMETERS	DL	DELAY-L DELAY AMOUNT TABLE
	DRL	DELAY-RL DELAY AMOUNT TABLE
	DRR	DELAY-RR DELAY AMOUNT TABLE
	FLP	TERMINAL FILTER-L PARAMETER
	FRP	TERMINAL FILTER-R PARAMETER
	MULTI1(M1)	TONE HOLE JUNCTION MULTIPLICATION COEFFICIENT 1
	MULTI2(M2)	TONE HOLE JUNCTION MULTIPLICATION COEFFICIENT 2
	MULTI3(M3)	TONE HOLE JUNCTION MULTIPLICATION COEFFICIENT 3
	MULTI4(M4)	TONE HOLE JUNCTION MULTIPLICATION COEFFICIENT 4
	J1	TUBE JUNCTION MULTIPLICATION COEFFICIENT 1
	J2	TUBE JUNCTION MULTIPLICATION COEFFICIENT 2
	J3	TUBE JUNCTION MULTIPLICATION COEFFICIENT 3
EG PAR	ATTACK RATE	ATTACK RATE
	RELEASE RATE	RELEASE RATE
	↓	↓
RESONATOR PAR	TYPE	RESONATOR TYPE
	FREQ	RESONATOR FREQUENCY CHARACTERISTIC PARAMETER
	LEVEL	RESONATOR LEVEL PARAMETER
	↓	↓
EFFECT PAR	EFFECT TYPE	EFFECT TYPE DESIGNATION
	FREQ	EFFECT DEPTH
	LEVEL	MODULATION SPEED
	↓	↓
SAMPLING FREQ	FS	SAMPLING FREQUENCY DATA (FS1>FS2)

FIG.29

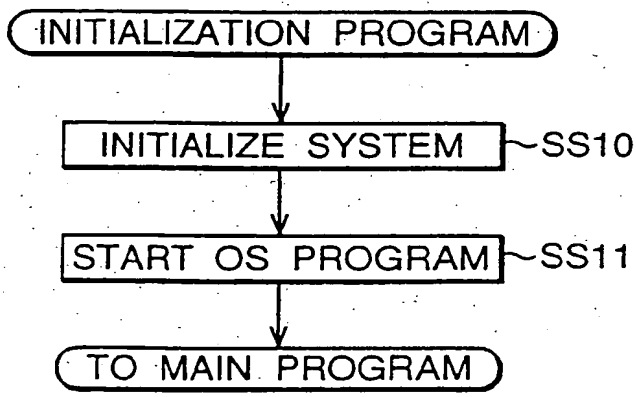


FIG.30

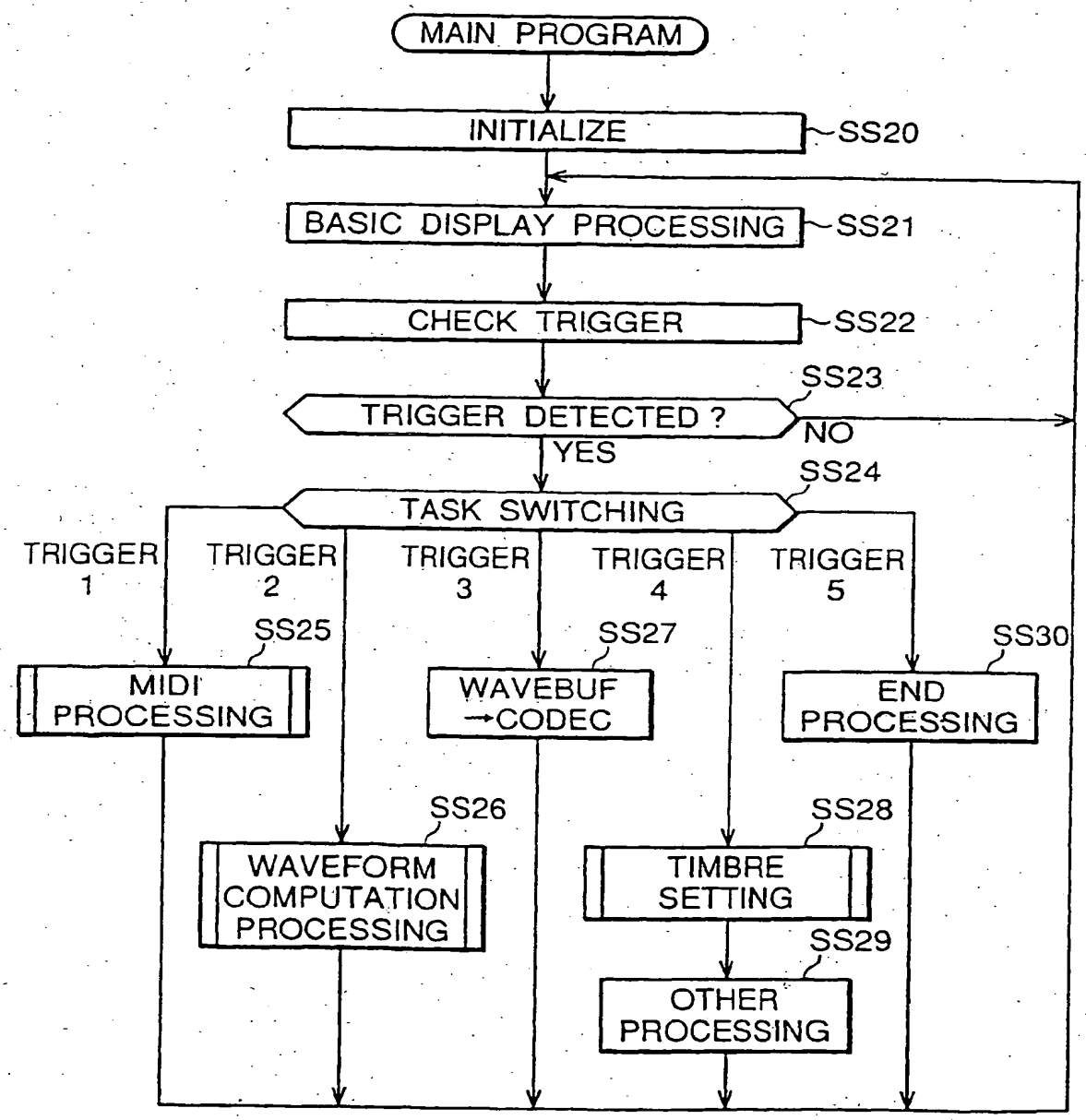


FIG.31

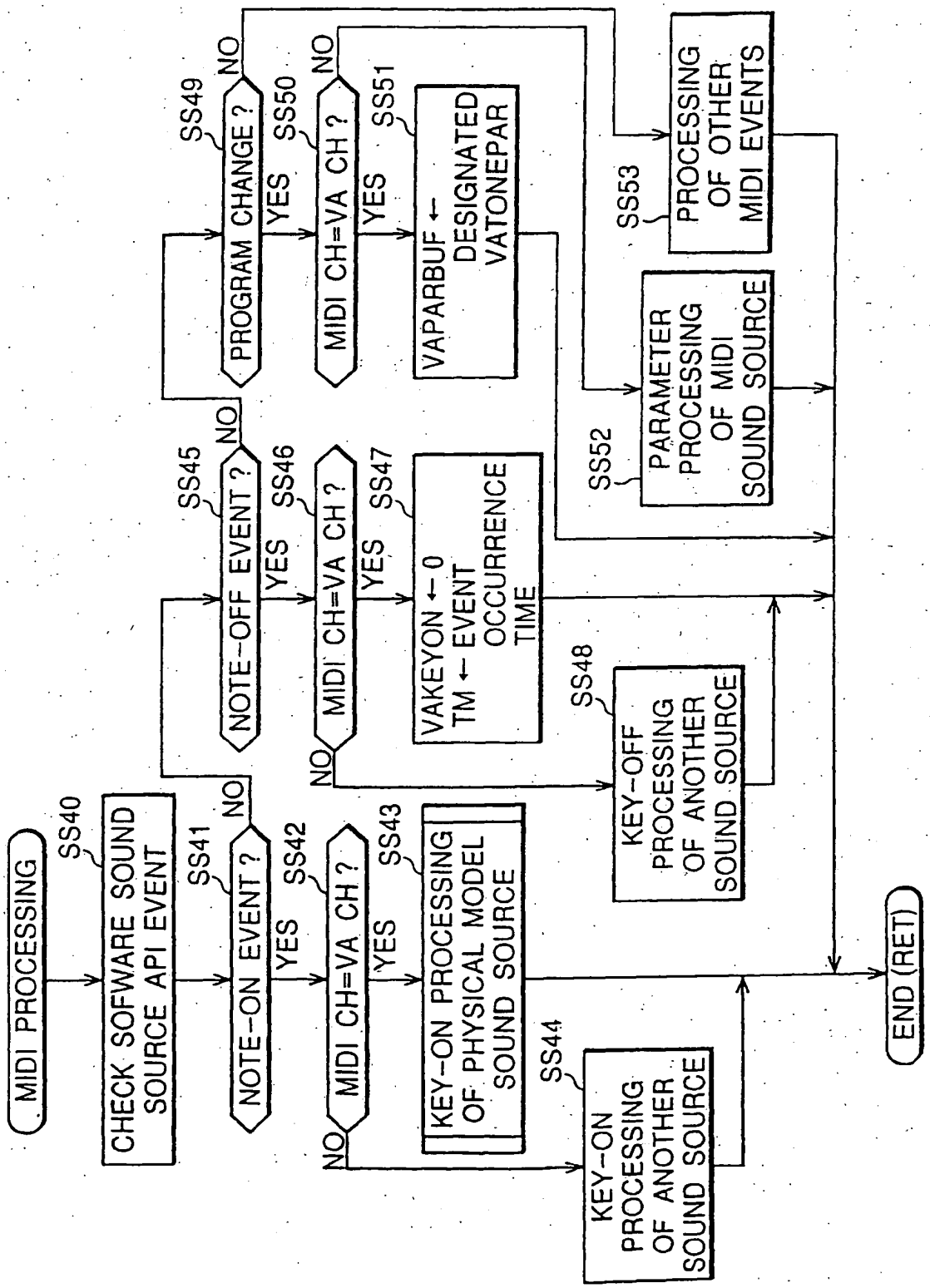


FIG.32A

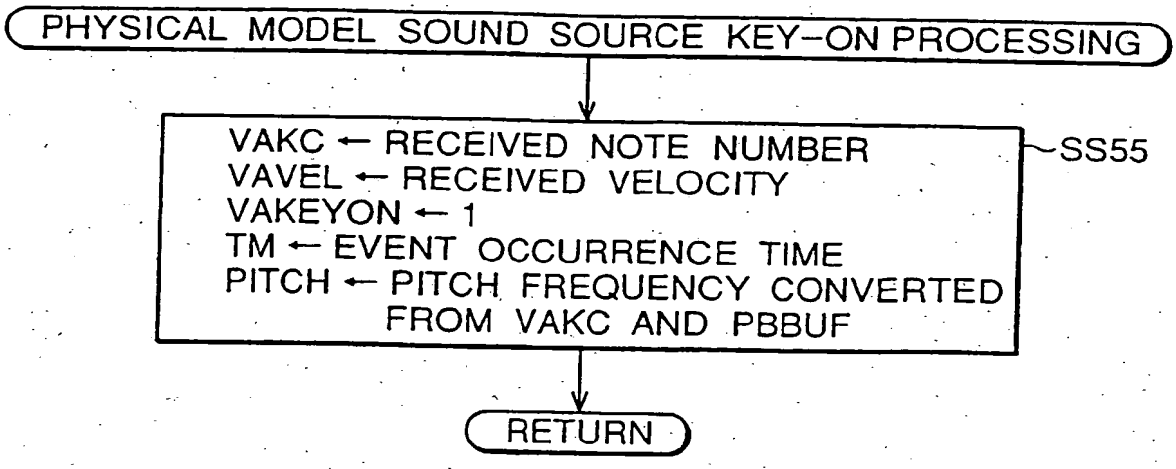


FIG.32B

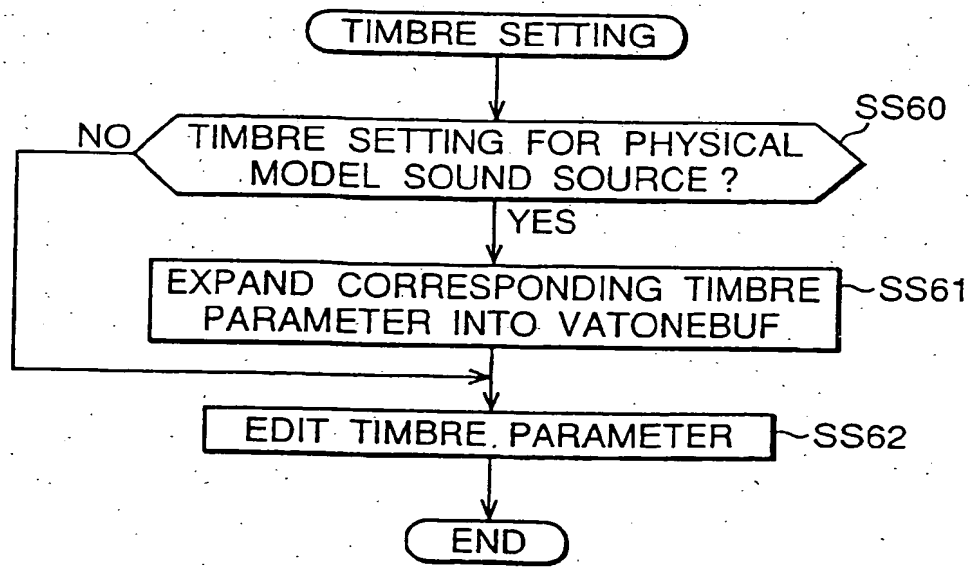


FIG.32C

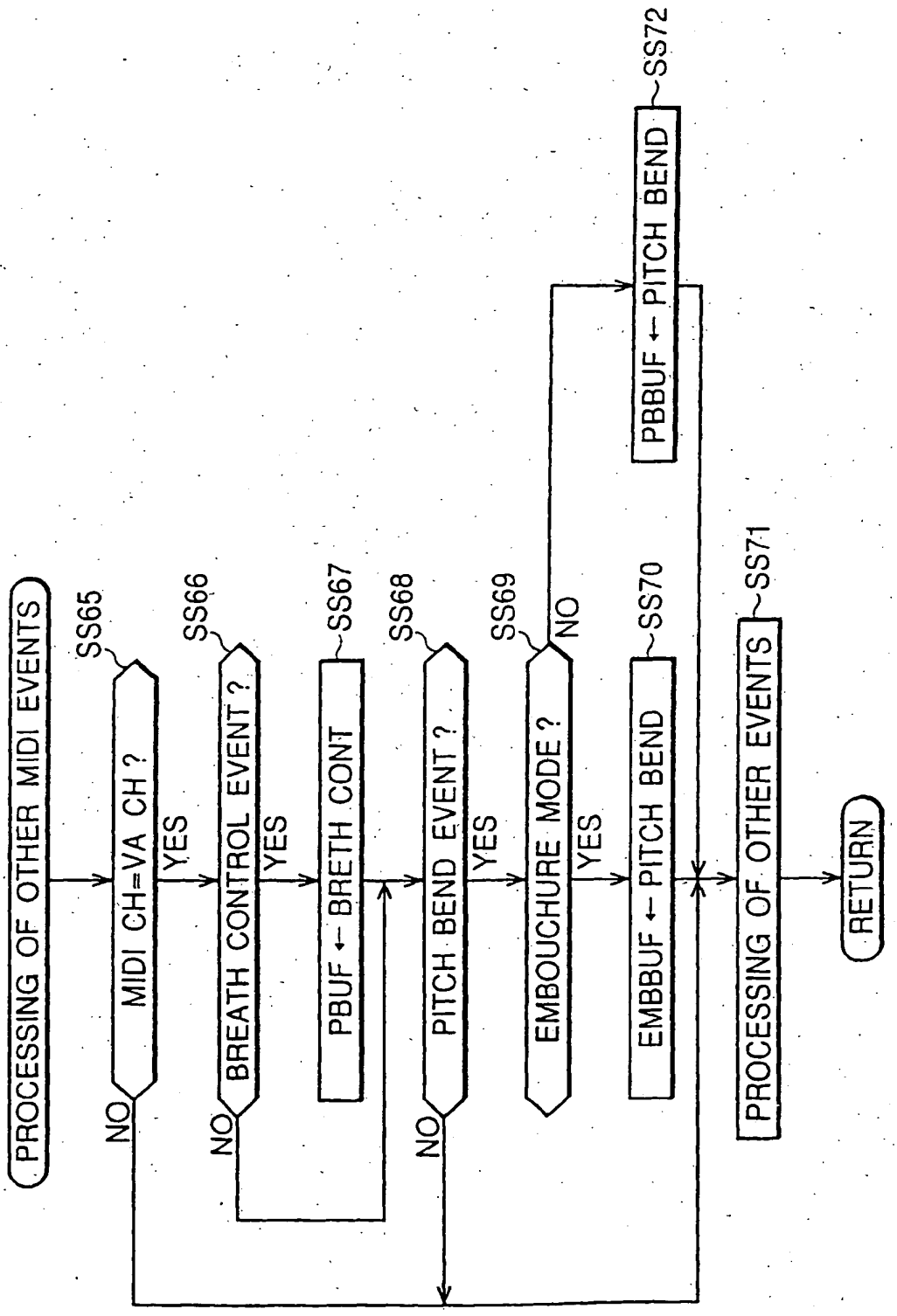


FIG.33

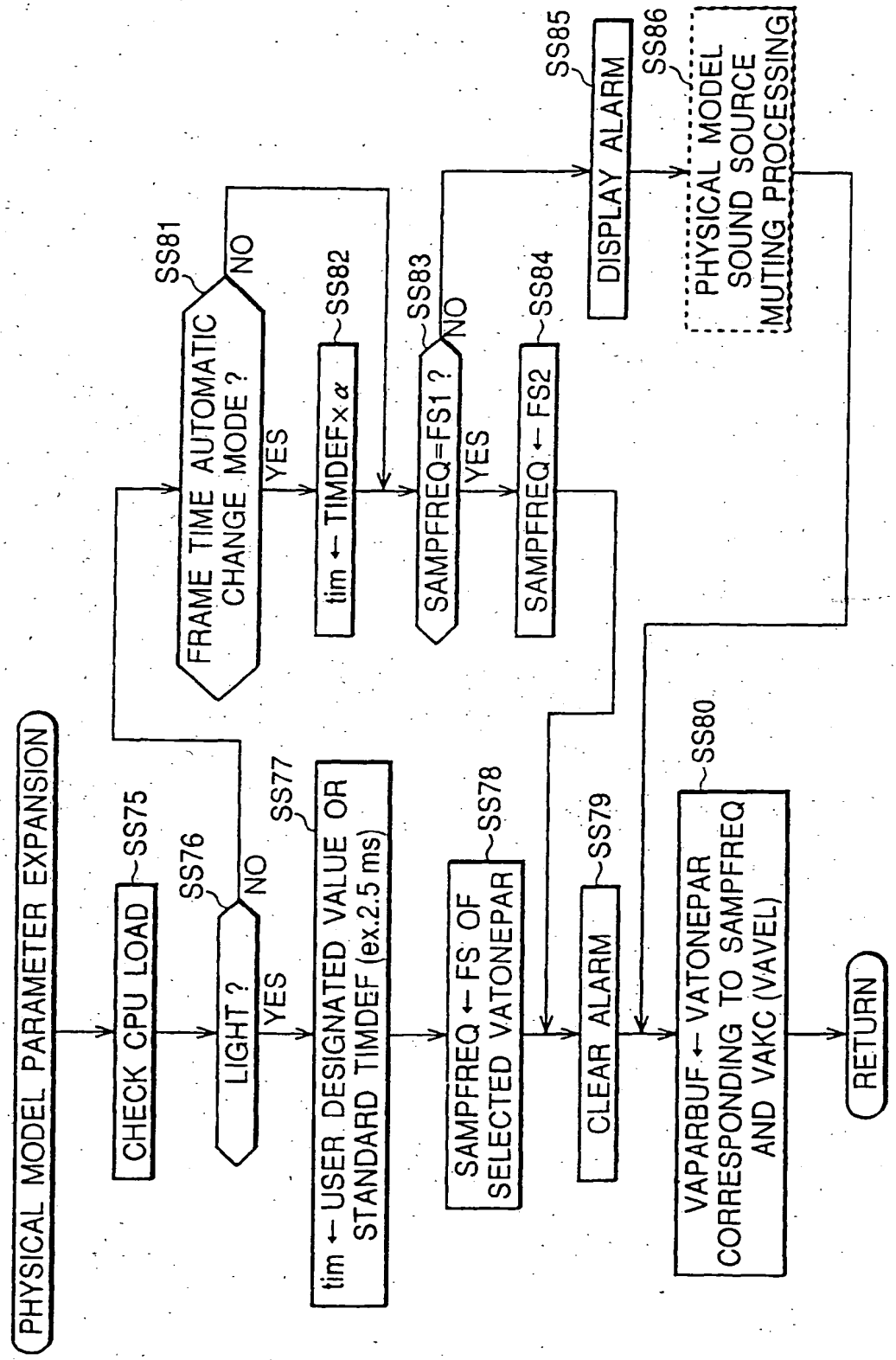


FIG.34

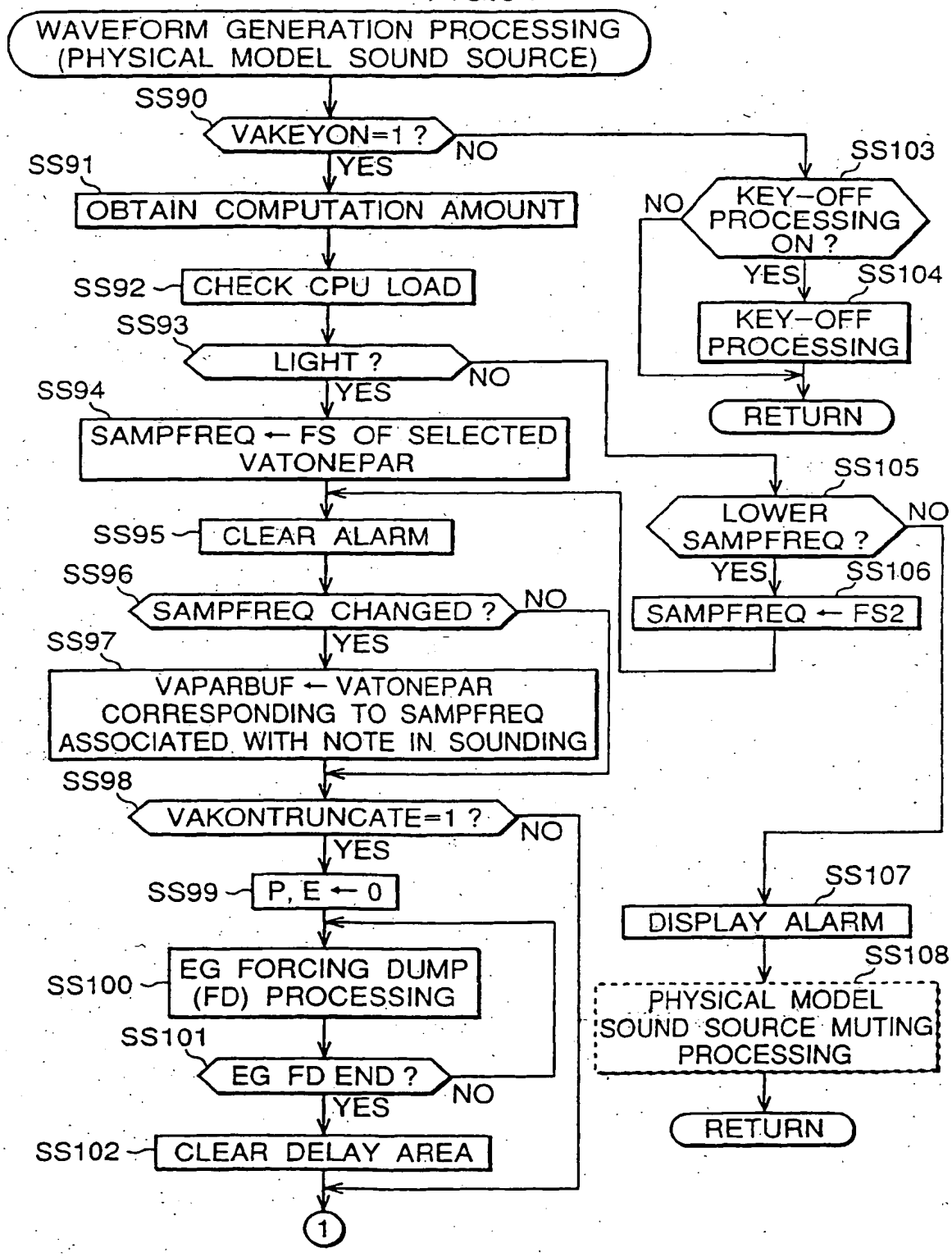


FIG.35

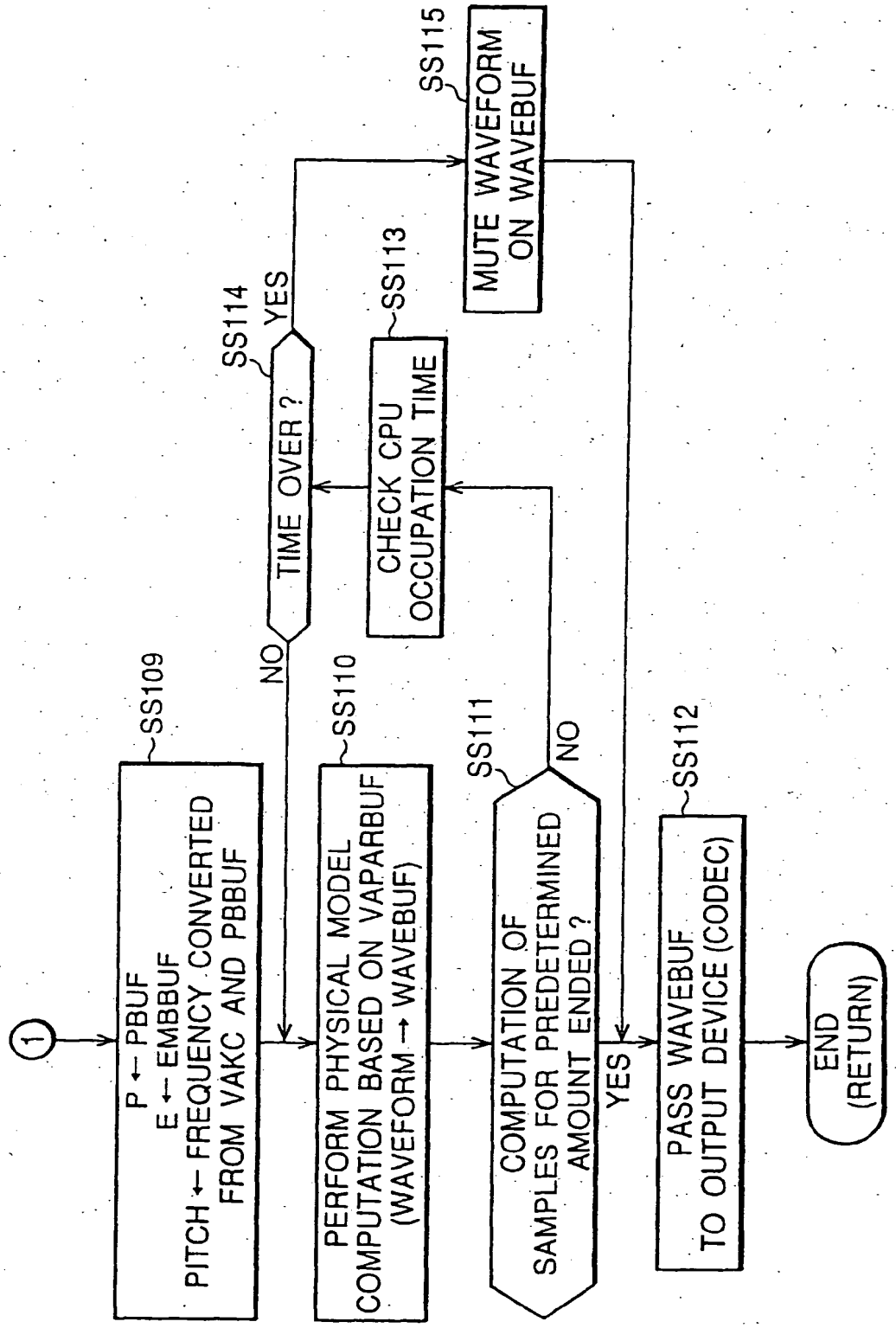


FIG.36

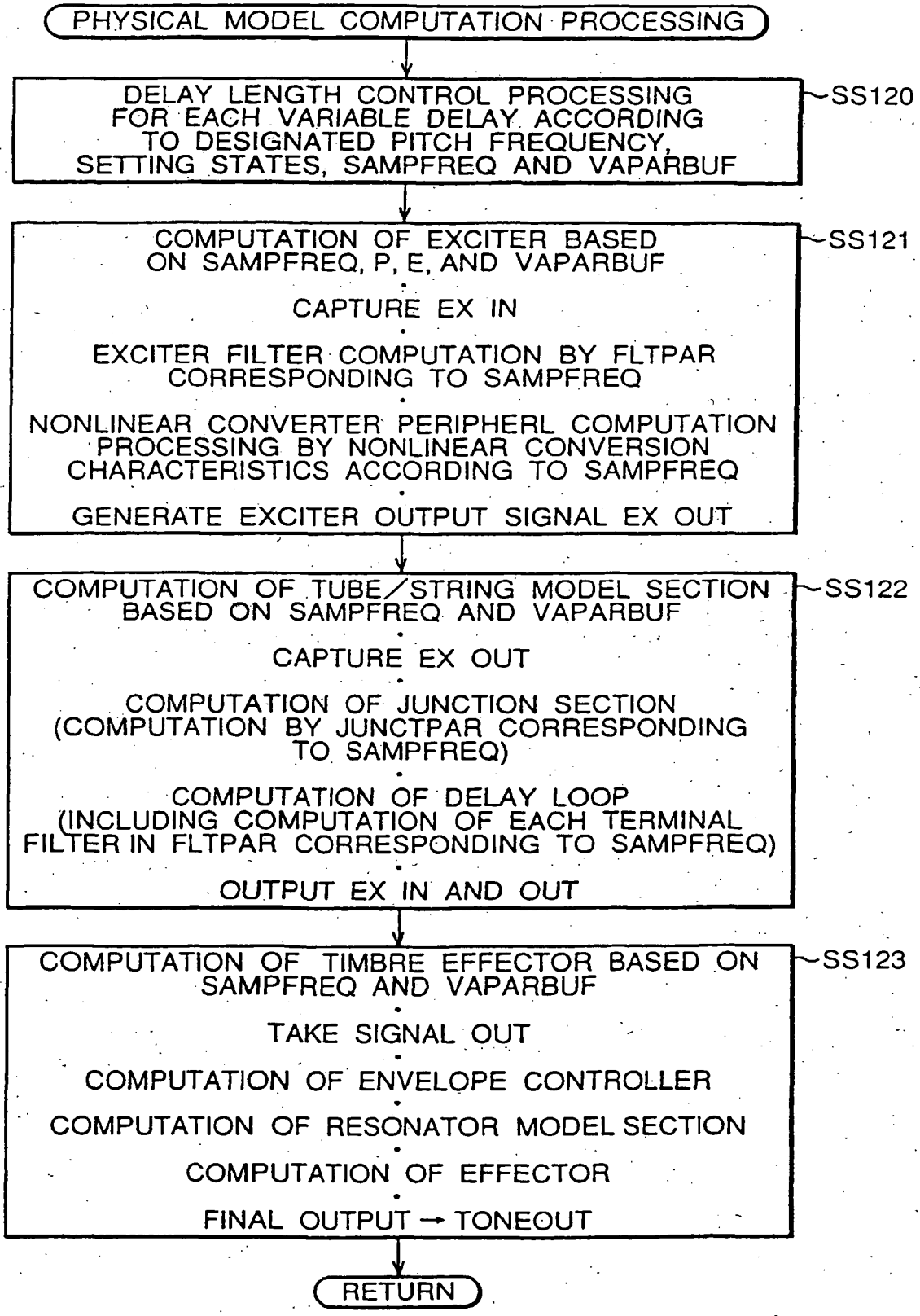


FIG.37

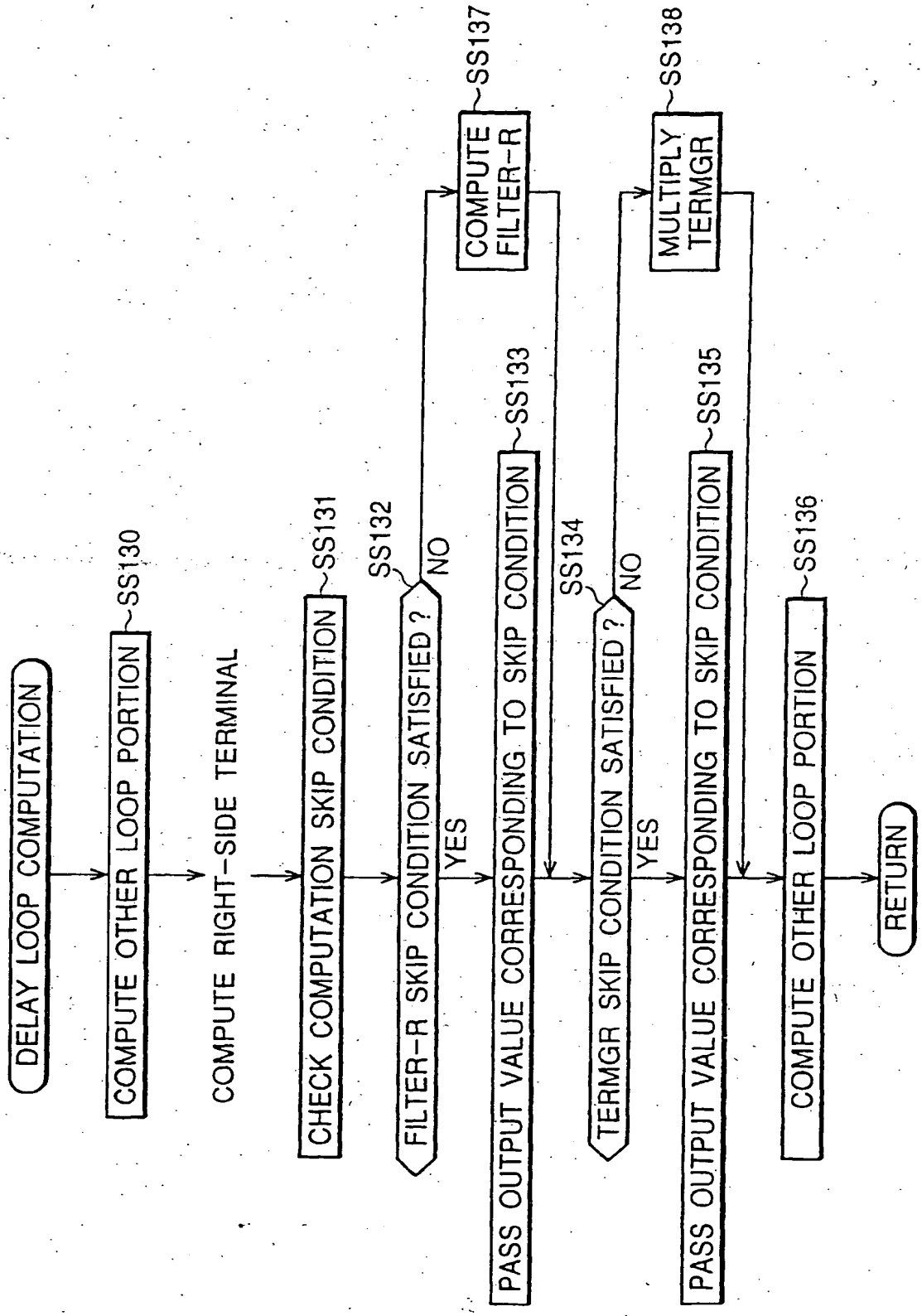


FIG.38

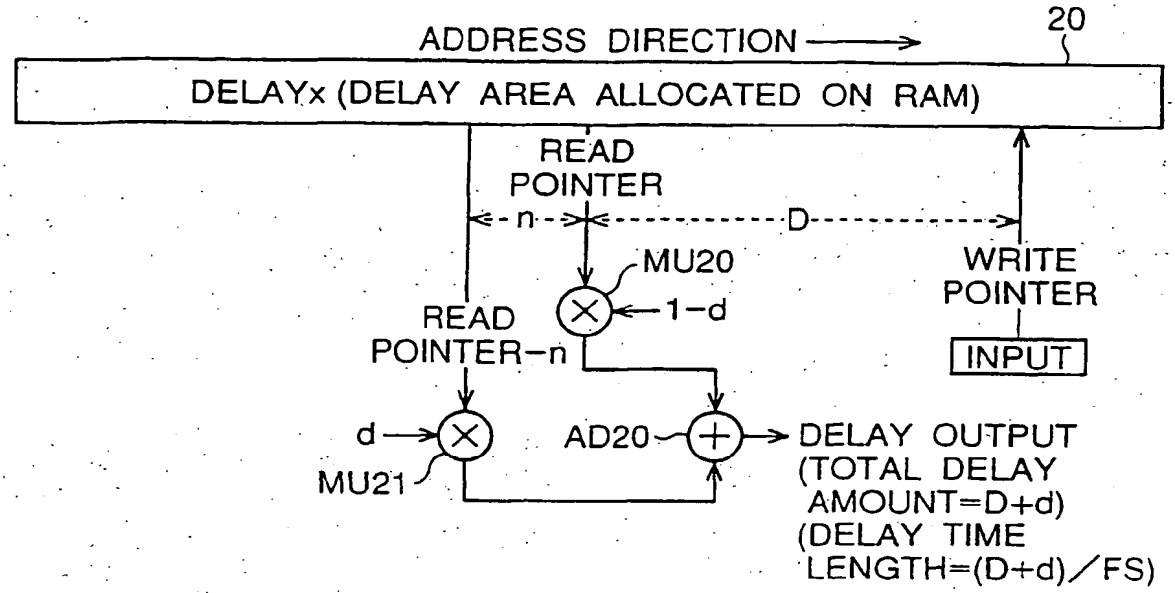


FIG.39

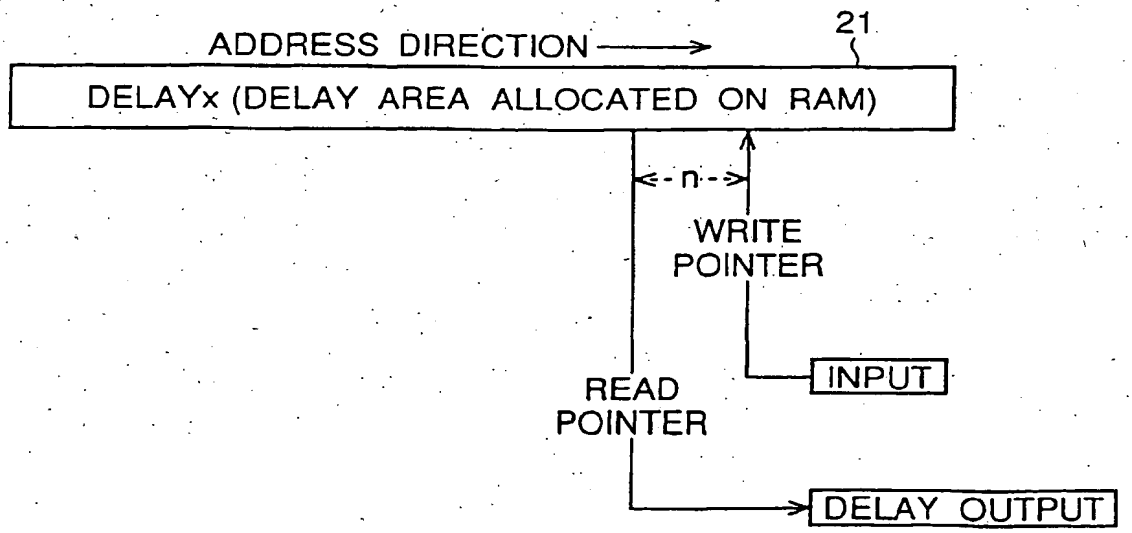


FIG.40A

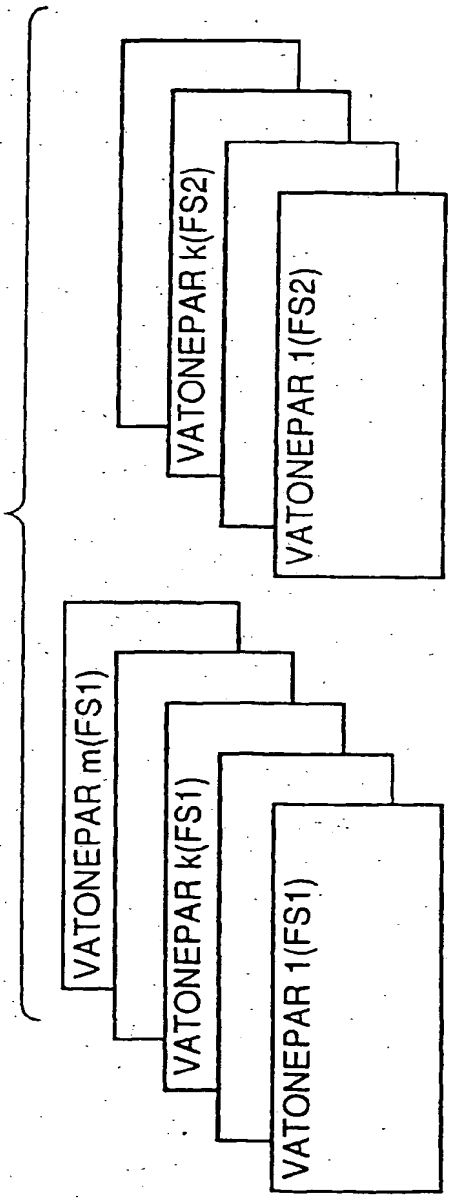


FIG.40B

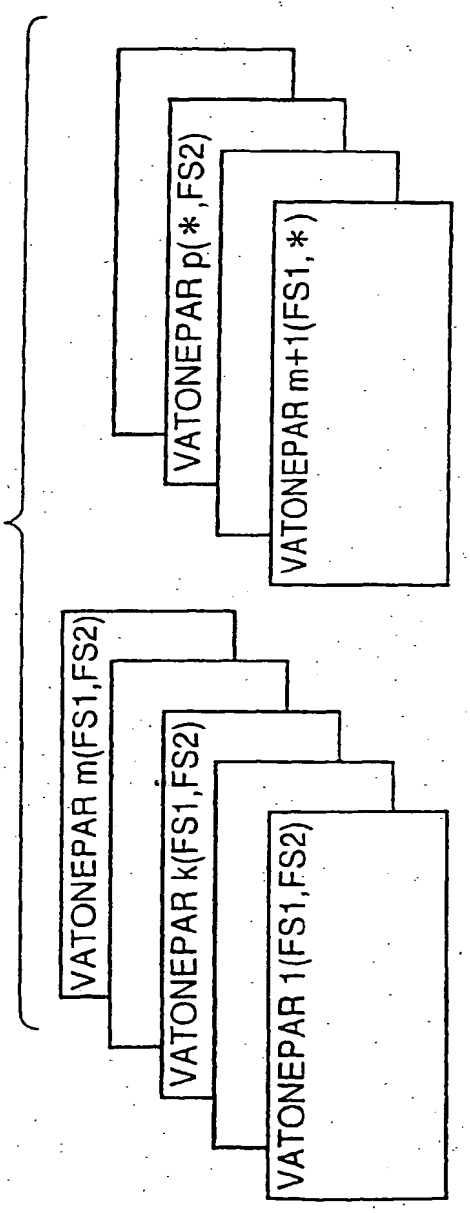


FIG.41

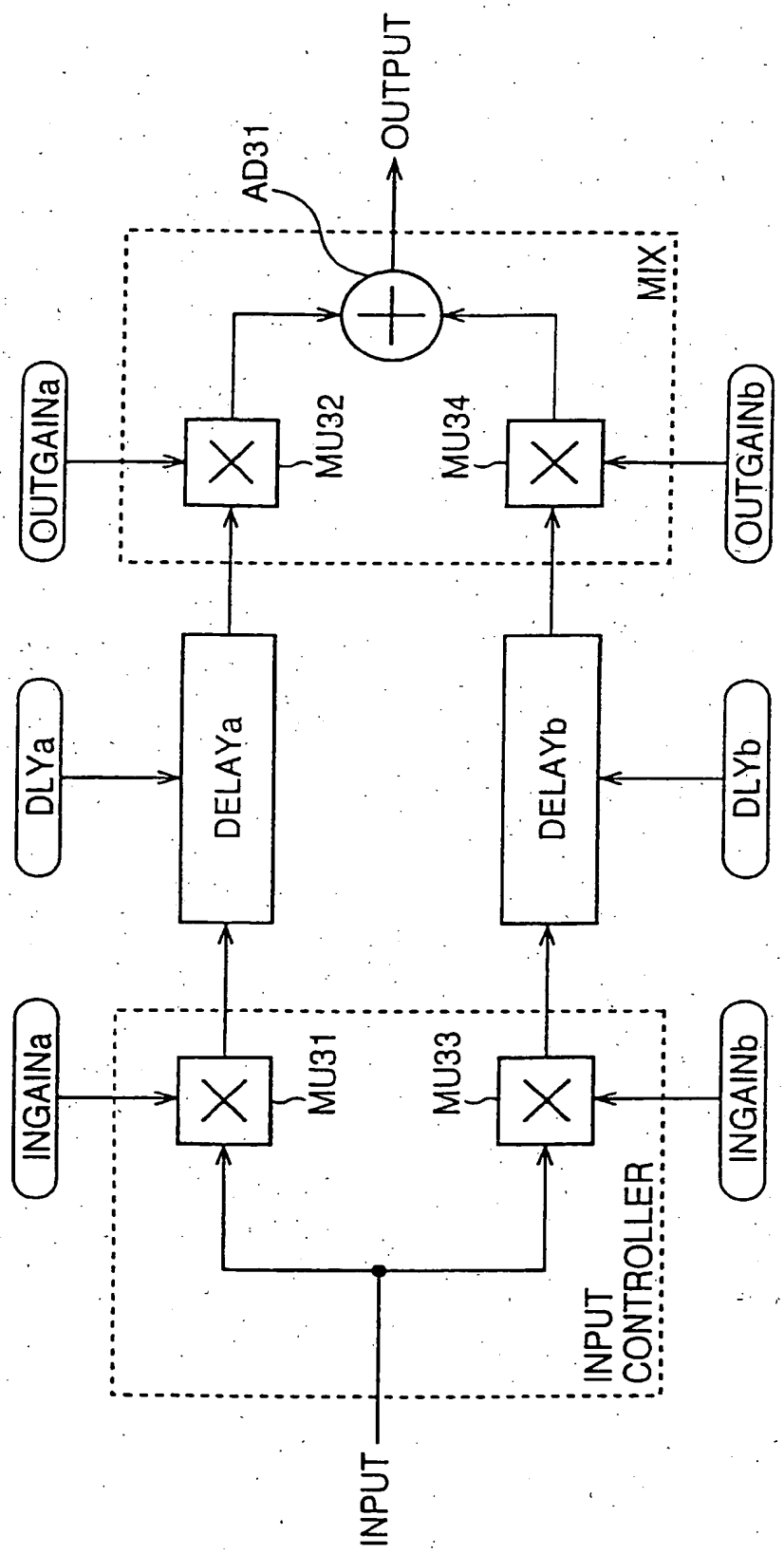


FIG.42A

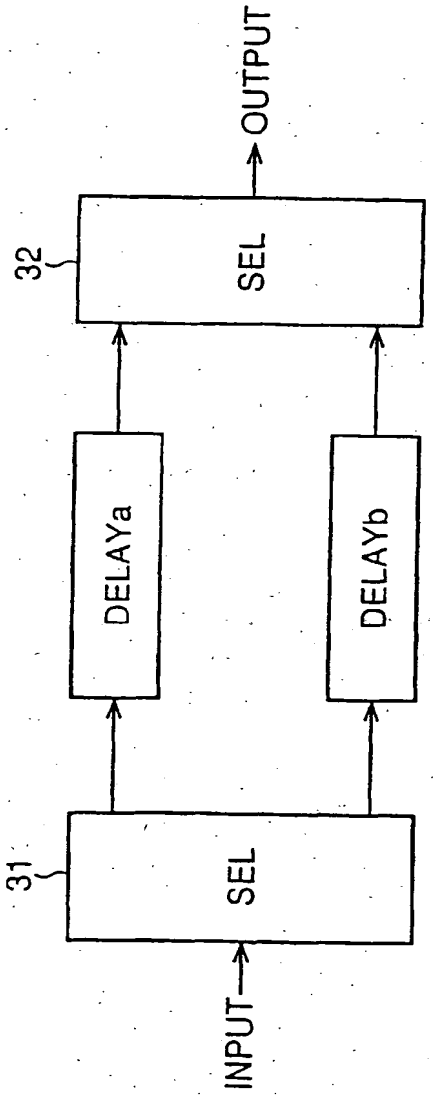


FIG.42B

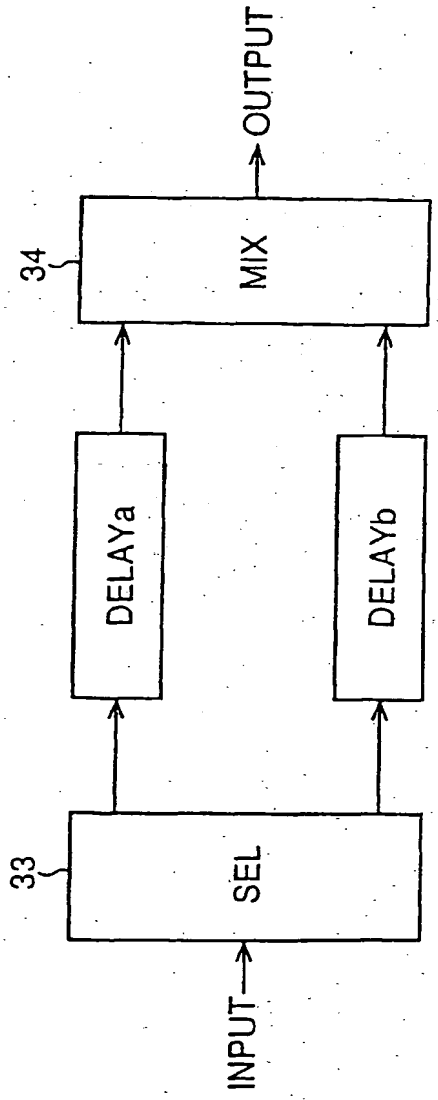


FIG.43

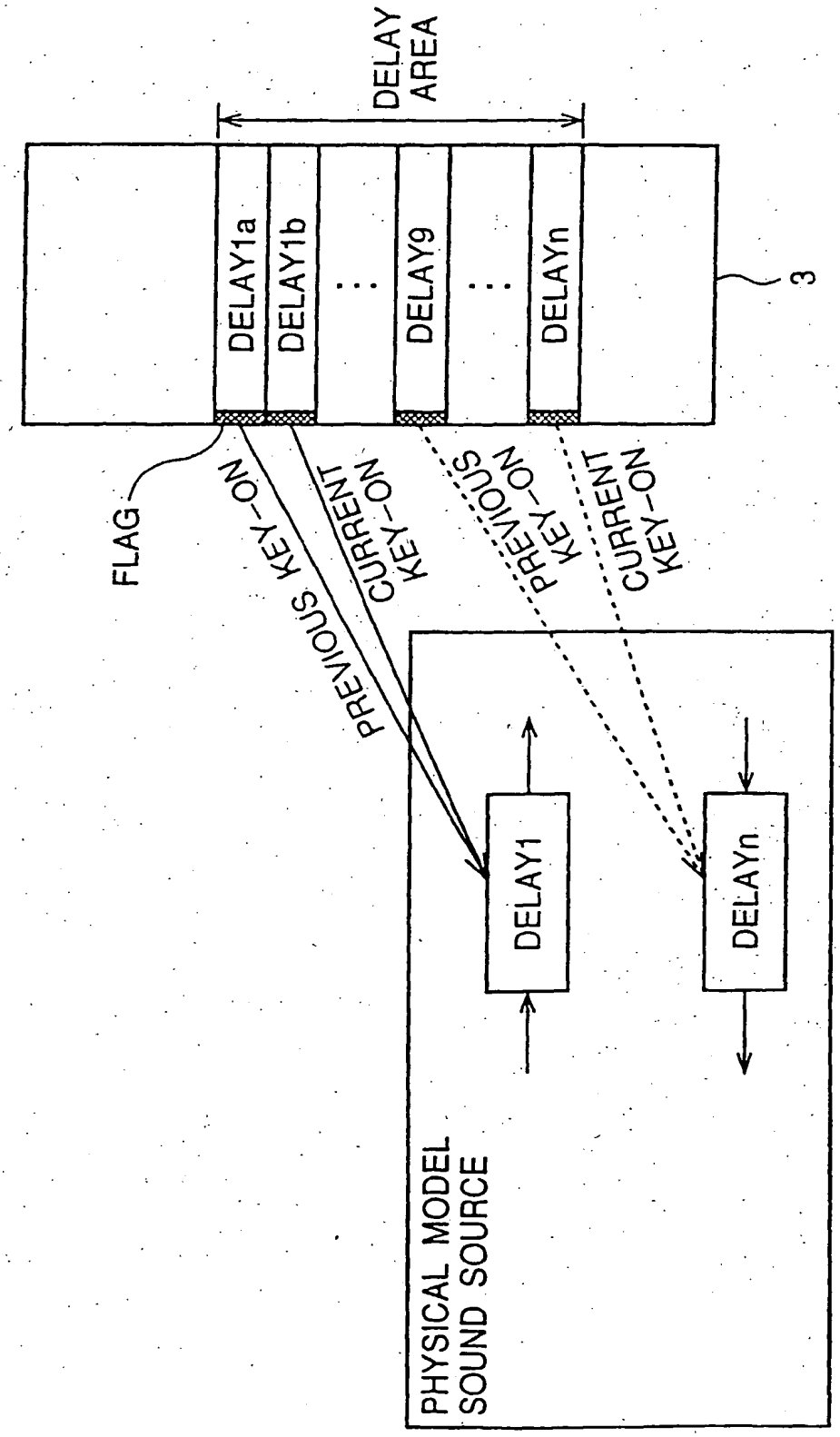


FIG.44